UNIVERSIDAD CARLOS III DE MADRID
Departamento de Ingeniería Telemática

PROYECTO FIN DE CARRERA

ESTUDIO DE LA ESTIMACIÓN DEL ANCHO DE BANDA DISPONIBLE EN COMUNICACIONES EXTREMO A EXTREMO

Autor: Manuel Antolín Ayuso

Tutor: Francisco Valera Pintor

Leganés, diciembre de 2010

Título: ESTUDIO DE LA ESTIMACIÓN DEL ANCHO DE BANDA DISPONIBLE EN COMUNICACIONES EXTREMO A EXTREMO

Autor: Manuel Antolín Ayuso
Director: Francisco Valera Pintor

EL TRIBUNAL

Presidente: Iván Vidal Fernández

Vocal: Julio Villena Román

Secretario: Matilde Sánchez Fernández

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día 15 de diciembre de 2010 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

Agradezco todo el apoyo que he recibido de mis familiares, novia y amigos. Sin todos ellos no hubiera podido realizar este sueño que tenía desde pequeño.

Ha sido una etapa dura de mi vida pero cargada de grandes momentos de felicidad e ilusiones.

Ahora espero emprender un camino lleno de posibilidades para aplicar lo aprendido durante estos años y obtener grandes éxitos.

Muchas gracias a tod@s.

Resumen

El presente proyecto fin de carrera tiene como objetivo el estudio de la estimación del ancho de banda disponible en comunicaciones extremo a extremo y desarrollar una aplicación en Java en la que se pueden probar varios algoritmos para obtener el ancho de banda disponible. De esta manera, mediante un simple Applet en JAVA, cualquier usuario podrá comprobar de forma rápida y sencilla el ancho de banda del que dispone además de poder realizar su propio estudio del ancho de banda en la red que desee medir.

En primer lugar, se realizará una descripción del estado del arte, en él se presentarán varias metodologías existentes para llevar a cabo la estimación del ancho de banda disponible (ABw). Con ello, se explicará el funcionamiento de los algoritmos.

Seguidamente, se presentarán los algoritmos desarrollados mediante la aplicación implementada.

Para terminar se presentarán los resultados obtenidos en algunos escenarios concretos.

Palabras clave: Java, Applet, Ancho de banda disponible (ABw), DP, ItP, TOPP, SLoPS, PathChirp, IP, TCP, UDP, Iperf, Redes, Protocolos de Comunicaciones.

Abstract

The purpose of this final project is to study the estimation of available bandwidth in end to end communications using different algorithms to develop a Java application to test them all. Thus, using a simple Java applet, anyone can quickly and easily check the available bandwidth, and also make its own study of bandwidth in the measured network.

First, there will be a description of the state of art in which several methodologies will be presented in order to get an estimation of available bandwidth (ABW). Also there will be explanations about the algorithms operations.

Next, the algorithms developed by the application deployed will be presented too.

Finally, the results obtained in some specific scenarios will be discussed.

Keywords: Java, Applet, Available Bandwidth (ABw), Available Bandwidth Estimated (ABwE), DP, ItP, TOPP, SLoPS, PathChirp, IP, TCP, UDP, Iperf, Networks, Communications Protocols.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción y objetivos

1.1 Introducción

En los últimos años se ha mostrado un gran interés en desarrollar técnicas para la estimación del Ancho de Banda disponible (*ABw*) a lo largo de una ruta de acceso a Internet.

La diversidad de rutas de acceso en redes superpuestas crea la necesidad de la estimación del ABw en estos caminos como método para elegir la mejor ruta. Además se puede asumir la cooperación de ambos, el remitente y el receptor, lo cual es necesario para la mayoría de las técnicas de medida.

El concepto de ancho de banda es fundamental para las comunicaciones digitales, y específicamente para redes de paquetes, ya que se refiere a la cantidad de datos que un enlace puede entregar por unidad de tiempo. Para muchas aplicaciones, como las transferencias de archivos o multimedia de “streaming”, el ABw para la aplicación afecta directamente en el rendimiento de las aplicaciones. Incluso para aplicaciones interactivas, las cuales son más sensibles a la latencia en lugar de querer obtener un mayor rendimiento, pueden beneficiarse de los bajos retardos producidos en los enlaces

de gran ancho de banda extremo a extremo y las bajas latencias existentes en la transmisión de paquetes [Rib+03].

El ancho de banda es un factor clave en varias tecnologías de red. Varias aplicaciones pueden beneficiarse y con ello mejorar su rendimiento si conocen las características de ancho de banda de su red. Por ejemplo, las aplicaciones peer-to-peer están basadas en redes con un ABw entre pares. Se pueden configurar redes superpuestas modificando las tablas de enrutamiento basándose en el ABw de los enlaces.

Los proveedores de redes alquilan a sus clientes sus redes basándose en el ancho de banda utilizado. Los acuerdos entre clientes y proveedores de servicio (SLAs) se suelen definir en términos de ABw en los principales puntos de interconexión (red frontera). Se puede establecer un plan de mejoras de la capacidad de su red en base a la tasa de crecimiento del ancho de banda producida por la utilización de sus usuarios.

El ancho de banda es también un concepto clave en las redes de distribución de sistemas de rutas de contenido inteligente, de control de admisión extremo a extremo, y de video / audio "streaming" [Rib+03].

El término ancho de banda es a menudo impreciso aplicado a una variedad de conceptos relacionados con el rendimiento. Podemos definir términos específicos relacionados con la métrica del ancho de banda, destacando el alcance y la pertinencia de cada uno. En concreto, en primer lugar, diferenciar entre el ancho de banda de un enlace y el ancho de banda de una secuencia de enlaces sucesivos a largo de una conexión extremo a extremo. En segundo lugar, hay que distinguir entre el máximo ancho de banda que un enlace puede disponer (Capacidad), el máximo ancho de banda no utilizado en un enlace (ABw) y el máximo rendimiento alcanzable en una conexión TCP (BTC). Todos estos indicadores son importantes desde diferentes aspectos del ancho de banda y son pertinentes para diferentes aplicaciones.

Por último, la cuestión más importante, ¿cómo obtener estos indicadores relacionados con el ancho de banda dentro de un enlace de red o en una conexión extremo a extremo? Un administrador de la red con acceso a un router o un switch conectado a un enlace de interés, puede medir algunos parámetros del ancho de banda directamente. Concretamente puede leer información asociada al router como por ejemplo parámetros de configuración, la tasa de bit por enlace, el promedio de utilización de bytes o paquetes transmitidos durante algún periodo tiempo, sin más que utilizando el protocolo de gestión de red SNMP. Sin embargo, este acceso está disponible sólo para los administradores y no para los usuarios finales. Los usuarios finales, por otra parte, sólo pueden estimar el ancho de banda de enlaces extremo a extremo, sin ningún tipo de información de los enrutadores de red.

Por ello el problema se va a centrar en el estudio de la realización de diferentes técnicas de medición de ancho de banda extremo a extremo sin causar una congestión añadida y apreciable durante la medida. Si bien todas las herramientas de estimación de ancho de banda van a tratar de identificar los *Cuellos de botella*[1]. De hecho, en algunos casos, no está claro si una metodología particular, en realidad mide el ancho de banda que se

[1] Cuello de botella: es el enlace más restrictivo de todo el camino.

pretende medir. Además, empleando metodologías similares, se pueden obtener resultados significativamente diferentes.

En este estudio se van a comparar diferentes metodologías e implementar algunas de ellas en una herramienta, comparando sus resultados experimentalmente. Se compararán dos de las técnicas más representativas como son Direct Probing (DP) techniques e Iterative Probing (ItP) techniques y en concreto se desarrollarán herramientas basadas en Iterative Probing technique, tales como Train Of Packet Pairs (TOPP), Self-Loading Periodic Streams (SLoPS) y PathChirp.

1.2 Objetivos

El objetivo fundamental del proyecto es desarrollar una herramienta web que permita determinar el ancho de banda del que dispone un usuario final en una comunicación extremo a extremo con distintos métodos y poder comparar los resultados obtenidos.

Para ello se ha realizado un estudio sobre diferentes metodologías con las que se va a implementar la aplicación. Con dicha aplicación se podrán probar los diferentes algoritmos diseñados para la obtención del Ancho de Banda disponible.

Se explicará el concepto de ancho de banda así como otros indicadores complementarios para dicha estimación.

Se aportarán soluciones para diversos escenarios y se propondrán diseños para futuros trabajos tales como la adaptación de la aplicación a otros entornos y arquitecturas.

1.3 Fases del desarrollo

Las Fases del desarrollo, han sido las siguientes:

Fase 1: localización de documentación y su estudio teórico.
Fase 2: estudio de las metodologías y selección de la más adecuada.
Fase 3: selección de los algoritmos más relevantes.
Fase 4: primera aplicación simple para comprobar su funcionamiento.
Fase 5: desarrollo de la aplicación.
Fase 6: pruebas en varios escenarios y obtención de resultados y validaciones.
Fase 7: elaboración de la memoria del proyecto.
Fase 8: presentación del proyecto ante el tribunal.

1.4 Medios empleados

Para la realización de este proyecto ha sido necesario contar con diversos recursos tanto hardware como software.

Debido a que se trata de un proyecto de desarrollo de un programa y además del tipo cliente-servidor el elemento hardware básico empleado es el ordenador. Para la realización completa del proyecto se han empleado varios equipos:

- Un ordenador principal de desarrollo, en que se ha desarrollado la aplicación, se ha ejecutado la parte cliente/servidor de la aplicación y se ha elaborado el documento definitivo.
- Un segundo ordenador en el que se ejecutado la parte servidor/cliente para probar la aplicación. También se ha utilizado para realizar algún desarrollo más.
- Un tercer equipo que se ha utilizado para generar tráfico cruzado. Además también se ha utilizado para ejecutar un emulador de red, lo que conlleva que disponga de dos tarjetas de red.
- Un cuarto equipo para poder observar y tomar muestras de cómo circulaban los paquetes por la red. Este equipo también disponía de dos tarjetas de red.

Los elementos software que se han empleado incluyen los siguientes programas:

- Windows 7 como sistema operativo principal [WIN7].
- Ubuntu 10.4 como sistema operativo en otros equipos [UBT].
- Java JDK 6 Update 21, como plataforma básica de desarrollo [JDK].
- IDE Eclipse (GanyMede) para realizar la labor de programación [ECS].
- WANem, sofware emulador de redes [WEM].
- D-ITG, para generar el tráfico cruzado [DITG].
- Jperf, software para medir anchos de banda utilizado para realizar validaciones. [JPF]
- MS Office Pro 2007 y MS Project Professional 2007, empleados para escribir la el documento final del proyecto y su presentación.
- Adobe Acrobat Professional 9.0 como herramienta de visualización de documentación utilizada en el proyecto y como medio para generar la versión PDF de la memoria.

1.5 Estructura de la memoria

Para facilitar la lectura del proyecto, se incluye a continuación un breve resumen de cada capítulo.

- *Capítulo 1*: en él se hace una breve presentación del proyecto en su conjunto, de la motivación inicial que sentó las bases del mismo, y de todos los procesos de análisis, diseño e implementación a que ha dado lugar.

- *Capítulo 2*: realiza una introducción sobre la medida del ancho de banda y los distintos métodos existentes para llevar a cabo dicha medición. Se introducen alguno conceptos necesarios para el posterior entendimiento así como también se añaden argumentos que ofrecen una pequeña motivación para la lectura del proyecto.

- *Capítulo 3*: presenta las diferentes técnicas y métodos estudiados para la realización del proyecto. Éste es el capítulo central de la tesis, ya que en él se basan los algoritmos implementados en la nueva herramienta que es el objetivo final del presente proyecto.

- *Capítulo 4*: en este capítulo se desarrolla la implementación de la aplicación, explicando la estructura de la aplicación a nivel de código y también como es su interfaz gráfica explicando brevemente su funcionamiento.

- *Capítulo 5*: presenta los resultados obtenidos. En este capítulo se muestran los diferentes escenarios en donde se ha probado la aplicación y sus correspondientes resultados.

- *Capítulo 6*: detalla los problemas encontrados durante la implementación y las posteriores pruebas realizadas.

- *Capítulo 7*: contiene el presupuesto asociado a la realización tanto de la aplicación como de la documentación asociada, detallando los costes materiales y de personal.

- *Capítulo 8*: presenta las conclusiones más importantes a las que se ha llegado como fruto de todos los procesos de análisis y diseño anteriores, así como del resultado de su implementación. Además incluye un conjunto de posibles líneas de investigación futuras que pueden servir para dar continuidad al presente proyecto.

- *Capítulo 9:* muestra las referencias de la bibliografía utilizada.

- *Apéndice 1*: corresponde a una pequeña guía de usuario en inglés de la aplicación.

- *Apéndice 2*: breve guía de usuario del emulador de redes WANem utilizado, también en inglés.

- *Apéndice 3*: pseudo-código de una parte de uno de los algoritmos, SLoPS.

- *Apéndice 4*: pseudo-código de una parte de otro de los algoritmos, PathChirp.

Capítulo 2

Estado del Arte

El crecimiento del número de usuarios de Internet, ha sido siempre un factor clave en el desarrollo tecnológico, ya sea de hardware o de software, para mejorar la interconexión del usuario a la enorme cantidad de servicios disponibles. Eso no habría sido posible sin estudios previos en el rendimiento de la red, como por ejemplo, el ABw. Entender las propiedades dinámicas del ABw extremo a extremo es beneficioso para una gestión adecuada de los recursos existentes y emergentes en los sistemas de comunicación. La tendencia creciente en la interfaz inalámbrica de datos significa que la tasa de los datos solicitada para un determinado servicio podría no ser garantizada, no sólo por la limitación de ancho de banda de la interfaz de aire, sino también por una limitación en el ABw en la red.

Algunos de los servicios de Internet actuales requieren supervisión del ABw, pero como se explica en la introducción, una medida directa mediante el despliegue de hardware o software en cada router de la red no sería ni eficiente ni profesional. Por esa razón, la investigación en la medición del ABw ha atraído un gran interés en los últimos años, y lo que ha llevado a una amplia variedad de métodos. Mediante la adaptación de estas técnicas, se podría obtener en tiempo real la estimación del ABw.

Este capítulo establece las bases de las técnicas de ABw. En primer lugar, se explican distintos conceptos de métrica de ancho de banda. En segundo lugar, se describe un modelo matemático para los retrasos en los paquetes en una red, que están directamente

relacionados con el ABw. Por último, se realiza un estudio de alguno de los métodos actuales más comunes de obtención del ABw [Pra+03].

2.1 Indicadores para la estimación del ancho de banda.

En este apartado, se definirán los indicadores que se pueden obtener o utilizar como métrica, tal y como se ha mencionado en la introducción. Estos indicadores son la Capacidad (*C*), el Ancho de Banda Disponible (*ABw*), el máximo rendimiento alcanzable de una conexión TCP (*BTC*) y por último se explicará un concepto llamado One-Way Delay, con el que se obtienen medidas del retardo de los paquetes.

2.1.1 Capacidad

La capacidad de un enlace se puede definir como la menor tasa de bit que es posible transmitir a lo largo de los segmentos individuales que se van encontrando en su ruta. La velocidad a la que un segmento de red puede transferir los datos es normalmente la tasa de transmisión o la capacidad del segmento. De esta forma, el enlace que determine la menor capacidad en el camino es el que determinará la capacidad de todo el enlace:

$$C = \min_{i=1\ldots H} C_i \tag{2.1}$$

Por otro lado, en un segmento o enlace, la capa de enlace puede transmitir datos a una tasa constate, por ejemplo, la tasa de un segmento 10BaseT Ethernet es de 10 Mbps o en un T1 es de 1.544 Mbps. Sin embargo, en la capa de red (IP), esta tasa siempre es menor debido a la cantidad de cabeceras que se introducen. Si el tiempo de transmisión para un paquete IP es:

$$T_{L3} = \frac{P_{L3}+O_{L2}}{C_{L2}} \tag{2.2}$$

donde P_{L3} es el tamaño del paquete IP, O_{L2} el tamaño de la cabecera del protocolo de nivel 2 (Ethernet, PPP,..) y C_{L2} es la capacidad del enlace a nivel de enlace. Si la capacidad a nivel 3 es:

$$C_{L3} = \frac{P_{L3}}{T_{L3}} = \frac{P_{L3}}{\frac{P_{L3}+O_{L2}}{C_{L2}}} = C_{L2}\frac{1}{1+\frac{O_{L2}}{P_{L3}}} \Rightarrow \frac{C_{L3}}{C_{L2}} = \frac{1}{1+\frac{O_{L2}}{P_{L3}}} \tag{2.3}$$

De esta manera, se pueden comparar dos protocolos de la capa de enlace como son PPP y Ethernet. El protocolo PPP tiene una cabecera que ocupa 8 bytes y la cabecera de Ethernet ocupa 38 bytes. Aplicando la fórmula expresada en (2.3) se obtiene el siguiente resultado (Figura 1).

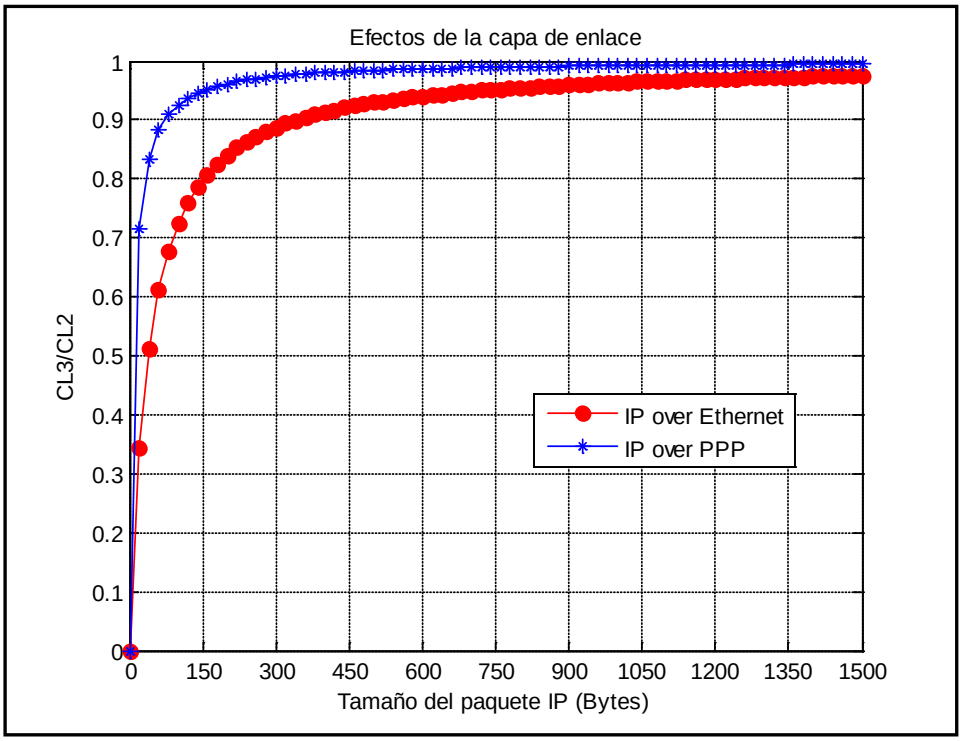

Figura 1. Relación entre capacidades

Se puede ver que tanto en los dos protocolos, si se utilizan paquetes IP menores de 200 bytes se comete un error bastante grande en la medida de la capacidad. Pero cuando se utilizan paquetes de tamaños más grandes, la capacidad calculada en cada capa se aproxima bastante. Esto nos quiere decir que hay que tener cuidado con el tamaño del paquete IP a la hora de medir la capacidad. También, hay que decir que el tamaño máximo que se ha determinado es de 1500 Bytes, puesto que es el MTU máximo que se puede transmitir en una red Ethernet.

Para finalizar, hay que destacar que existen otras tecnologías de nivel 2 que no transmiten a una tasa constante, como es el caso de las redes que utilizan la tecnología Wireless IEEE 802.11b. En este caso se utilizan transmisiones de 11, 5.5, 2 ó 1 Mbps en función de la tasa de error encontrada en dicha transmisión. La primera definición de capacidad que se utilizó en (2.1) puede aplicarse en estas tecnologías siempre y cuando se utiliza en un intervalo de tiempo en el que se esté transmitiendo a una tasa constante [CV06].

2.1.2 Ancho de Banda disponible(*ABw*)

El indicador más importante en este estudio es el *ancho de banda disponible* (*ABw*) de un enlace extremo a extremo. El *ABw* de un enlace se refiere a la parte no utilizada de la capacidad total del enlace durante un cierto período de tiempo. Por lo tanto, aunque parezca que la capacidad de una conexión depende de la tasa de transmisión de la tecnología empleada y del medio de propagación utilizado, además, depende de la carga de tráfico que haya en ese enlace que variará con el tiempo.

Puesto que en cualquier instante de tiempo, puede surgir una nueva conexión dentro del enlace, para poder medir correctamente este indicador hay que realizar las medidas del

ancho de banda en un intervalo de tiempo sobre el cual se hará un promedio. Esto se puede expresar mediante la siguiente ecuación:

$$\bar{u}(t-\tau, t) = \frac{1}{\tau}\int_{t-\tau}^{t} u(x)\, dx \tag{2.4}$$

Donde *u(x)* es el *ABw* en un instante de tiempo determinado *x*.
Por ejemplo, en la Figura 2 podemos ver la utilización de un enlace en un intervalo de tiempo T.

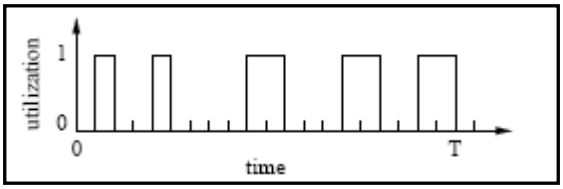

Figura 2. Uso instantáneo de un enlace en un intervalo de tiempo (0, T) [Pra+03]

Se puede ver que el enlace se ha utilizado durante 8 de los 20 intervalos de tiempo, de forma que, en media, se ha utilizado un 40% de su capacidad. Ahora bien, se puede calcular el *ABw* en un segmento, de forma que si C_i es la capacidad del segmento *i*, u_i es la utilización media de dicho segmento en un intervalo de tiempo dado, el valor medio del *ABw* A_i se pude expresar de la siguiente forma:

$$A_i = C_i(1 - u_i) \tag{2.5}$$

De la misma forma que la capacidad, el *ABw* será el mínimo encontrado a lo largo de un enlace o de varios segmentos:

$$A = \min_{i=1\ldots H} A_i \tag{2.6}$$

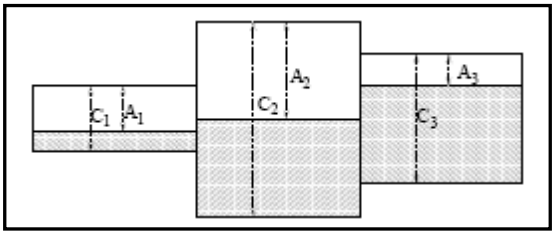

Figura 3. Tráfico cursado en tres segmentos de la red [Pra+03]

En la Figura 3 se puede ver como el último segmento tiene el menor *ABw* (A_3) y va a ser el *cuello de botella* de la transmisión en ese instante de tiempo.
Es importante destacar que en muchas ocasiones se asume que la carga de tráfico es estacionaria en todo el camino. Esto sólo es razonable tomando un intervalo corto de tiempo puesto que es un indicador que varía rápidamente con el tiempo. Este hecho es la principal diferencia que hay con respecto a la capacidad, puesto que no cambia tan rápido mientras no existan modificaciones en las rutas o en los enlaces [Pra+03].

2.1.3 One-Way Delay (*OWD*) [CV06]

Con el objetivo de entender los distintos métodos de estimación de ancho de banda, es necesario desarrollar una descripción matemática de los retrasos que experimentan los paquetes desde el salen del origen hasta que llegan a su destino.

Se define One-Way Delay (OWD) como el retardo que experimenta el paquete en el camino de ida, es decir, el tiempo que utiliza un paquete k para llegar a su destino. Este retardo depende del tiempo de transmisión, la latencia y el retardo en cola. El tiempo de transmisión es tiempo empleado por el router para transmitir un paquete, que es función del tamaño del paquete y la capacidad de la conexión. La latencia de cola es el tiempo que emplea la señal para atravesar el enlace, determinada por las características físicas del enlace. El retardo de cola es el tiempo que tiene que esperar un paquete en el router debido al tráfico cruzado. Los dos primeros términos son deterministas mientras que el último es aleatorio. Por lo tanto, el OWD se puede expresar como:

$$\Omega_k^h = \sum_{s=1}^{h}(x_s + d_s + q_s) = \sum_{s=1}^{h}\left(\frac{P_k}{C_s} + d_s + q_s\right) \tag{2.7}$$

donde x es el tiempo de transmisión de un paquete del tamaño de P_k, d_s es la latencia de cola y q_s es el retardo de cola. Para medir el OWD, es necesario tener marcas de tiempo, tanto en el origen y como en el destino. Para algunas aplicaciones puede ser interesante una única medida en el origen utilizando *Round-Trip Time (RTT)*, que es el tiempo que tarda en ir y volver un paquete a lo largo del camino. Esto se puede expresar como:

$$\mathrm{RTT}_k^h = \sum_{s=1}^{h}\left(\frac{P_k}{C_s} + d_s + q_s\right) = \sum_{s=h}^{1}\left(\frac{P_r}{C_s} + d_s + q_s^r\right) \tag{2.8}$$

2.1.4 Máximo rendimiento en una transmisión TCP (*BTC*)

Otro indicador relativo al ancho de banda es el rendimiento de una conexión TCP/IP. TCP es el protocolo de transporte más importante que existe en Internet, su uso es casi el 90% del tráfico. Por tanto, obtener una medida de su rendimiento sería de gran interés para los usuarios finales.

Lamentablemente, no es fácil obtener el rendimiento de una conexión TCP. Existen varios factores que pueden influir en el rendimiento de TCP, como pueden ser el tamaño de las transferencias, el tipo del tráfico cruzado (UDP o TCP), el número de competidores de las conexiones TCP, el tamaño de la ventana inicial, etc.

Por ejemplo, transferencias tales como la de una típica página web dependen principalmente de la primera ventana de congestión, tiempo de ida y vuelta (RTT) y el mecanismo de arranque de TCP "*Slow-Start*[2]", en lugar de tener en cuenta el ancho de banda de la ruta. Además, el rendimiento de transferencia de TCP puede variar

[2] *Slow Start*: algoritmo para el cálculo de la ventana de congestión aplicado al principio de la conexión, y hasta que se alcanza el umbral de congestión.

significativamente cuando se utilizan diferentes versiones de TCP, incluso si el ABw es el mismo.

Bulk Transfer Capacity (BTC) define un indicador que representa el rendimiento alcanzable para una Conexión TCP, es decir, BTC es obtener el máximo rendimiento por una sola conexión TCP. En la conexión se deben poder aplicar todos los algoritmos de control de congestión TCP tal como se especifica en el RFC 2581. Sin embargo, esta RFC deja abiertos algunos detalles de implementación, por lo que una medida también debe especificar en detalle otros parámetros importantes acerca de la aplicación (o emulación) de TCP.

Hay que tener en cuenta que el ABw y BTC son parámetros diferentes. BTC es específico para una conexión TCP, mientras que el ABw no depende de un protocolo de transporte. El BTC depende de cómo se comparta el ancho de banda con otras conexiones TCP, mientras que el ABw asume que el promedio de carga de tráfico se mantiene constante y estima el ancho de banda que queda disponible en el enlace.

Para ilustrar el funcionamiento de BTC, se puede suponer un enlace con capacidad C que este saturado por una conexión TCP. Si se quiere calcular el ABw, el resultado obtenido será cero, puesto que el enlace está saturado y no se podrán llevar a cabo más conexiones hasta alguien deje de transmitir. Sin embargo con BTC, se modificarían los parámetros de TCP y se reduciría la tasa de transferencia a la mitad (C/2) para que otra conexión TCP se pueda incorporar al enlace sin que cambie nada [Pra+03].

2.2 Modelos actuales de medida

Los tests de velocidad actuales, por razones de simplicidad, basan su estimación en medir el tiempo que el usuario necesita para descargar un fichero de tamaño conocido a través de una conexión TCP. Así, dividiendo el tamaño del fichero por el tiempo invertido en su descarga, se determina el caudal medio que tuvo esa conexión y este valor se emplea como estimador del ancho de banda disponible en el acceso del usuario. Para facilitar la realización del test, el proceso de descarga se oculta al usuario tras una interfaz sencilla (en flash o Java), que sirve luego para presentar los resultados de la prueba de una manera amigable.

Una variante de esta metodología es la del llamado “Test de Ookla”, donde se toman númerosas muestras a lo largo de la descarga (contando los bytes transferidos en cada intervalo), se descartan muestras “extrañas”, y al final se hace un promedio de todas esas estimaciones parciales. Esta variante, que permite ir ofreciendo al internauta “resultados intermedios” a modo de velocímetro, es la que emplean gran parte de las webs españolas que ofrecen medidores de velocidad [Ram08].

Hay que señalar que esta metodología se centra, por tanto, en estimar el ancho de banda que “llega” a la aplicación y no el realmente provisto por la red (que incluiría el transporte de todas las cabeceras de enlace, de IP, de TCP, etc.). Si bien la diferencia entre ambas magnitudes puede llegar a ser de algunos puntos porcentuales, como primera aproximación suele considerarse ésta una fuente de error “controlada”, ya que es de un

orden de magnitud relativamente modesto y, casi siempre, es constante en porcentaje respecto a la capacidad del acceso que se quiera medir. Sin embargo, para que lo anterior sea cierto, es necesario garantizar que el usuario no está utilizando ninguna otra aplicación durante la realización de la prueba (por ejemplo, descargándose un fichero al mismo tiempo). Esta condición será especialmente difícil de verificar completamente si se trata de un test de velocidad accesible al gran público a través de Internet.

No obstante, la principal fuente de error en tests basados en esta metodología procede del impacto que el propio protocolo TCP y sus mecanismos de control de flujo tienen en la velocidad de descarga cuando la red de acceso es de alta capacidad. Así, factores como la configuración de TCP en el servidor de medida y en el host del usuario, la latencia entre ambos extremos o el tamaño de la descarga, pueden tener un impacto determinante en el máximo caudal que puede alcanzarse en la conexión TCP de la prueba y, por lo tanto, en la velocidad que finalmente se estima para ese acceso. Uno de los principales problemas es cuando se establece una conexión TCP, si existen otras conexiones TCP en ese camino, esa nueva conexión afectará al rendimiento de todas las conexiones, incluida la nueva. Además de influir, lo que se está midiendo ya no es el ABw, si no el ancho banda que te han dejado para establecer una conexión TCP, lo que hemos denominado BTC [Ram08].

Otro punto importante es el grado de intrusión en la red a la hora de realizar la medida. Con esta metodología, al enviar un fichero, el cual tiene que ser lo suficientemente grande para que la medida sea lo bastante fiable, hace que la red tenga un carga añadida y esto hace que la medida no sea del todo precisa, además de congestionar la red.

Por todo ello, se han realizado diversos estudios para la obtención del ABw sin que el propio proceso de medida afecte de una forma elevada a la estimación del ABw, ya sea tanto en que las demás conexiones no se enteren del proceso de medida, como en que éste no sea demasiado intrusivo para la red.

Debido a estas condiciones, para poder desarrollar las nuevas metodologías se utiliza el protocolo UDP en vez de TCP puesto que es mucho más ligero y sus propiedades ofrecen grandes ventajas para poder llevar a cabo la obtención del ABw.

2.3 Diferentes técnicas para obtener el ABw

Para introducir el estudio y poder entender mejor las técnicas implementadas en este proyecto, se van a describir dos técnicas, Direct Probing (DP) que están basadas en Probe Gap Model (PGM) e Iterative Probing (ItP) que se basan en Probe Rate Model (PRM), ya mencionadas anteriormente. También se describirán brevemente unas técnicas mixtas en las que se utilizan dichas técnicas, que se han desarrollado en una herramienta llamada Initial Gap Increasing (*IGI)* [HS03].

En este estudio sólo se han desarrollado las ItP, pero es interesante mostrar otro tipo de técnicas existentes y poder establecer una pequeña comparación entre ellas, además de dar una visión más general al estudio en cuestión.

Dichas técnicas han sido clasificadas dependiendo de los mecanismos con los que se realiza la toma de muestras y cómo analizarlas, aunque como veremos, son complementarias y se pueden mezclar.

Suponiendo que la tasa de tráfico cruzado es constante para un intervalo de tiempo dado, el ABw se puede obtener a partir de la ecuación 2.9:

$$A = C_t - R_x \tag{2.9}$$

donde C_t es la capacidad del enlace que produce el cuello de botella y R_x es el tráfico cruzado que atraviesa el enlace.

El funcionamiento básico de estas técnicas es el siguiente:

Cuando se está enviando a través de un enlace una tasa de envío superior a la disponible, esto es $R_{in} > A$, dicho enlace estará sobrecargado y con esto se generará una tasa de sobrecarga denominada R_{ov}, que será un fracción de la tasa de envío inicial que el enlace no podrá manejar (Figura 4) [CV06]:

$$R_{ov} = R_{in} - A \tag{2.10}$$

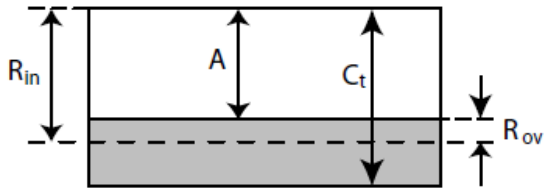

Figura 4. Representación gráfica de la tasa de sobrecarga R_{ov} [CV06]

De esta forma, si el tiempo entre dos paquetes consecutivos se denomina:

$$T_{in} = \frac{P}{R_{in}} \tag{2.11}$$

donde P es el tamaño del paquete; durante dicho tiempo el enlace no procesará $R_{ov}T_{in}$ bytes y se producirá un retardo de cola Δq de la forma:

$$\Delta q = \frac{T_{in}R_{ov}}{C_t} = \frac{P}{C_t}\left(\frac{R_{in}-A}{R_{in}}\right) \quad \text{si } R_{in} > A \tag{2.12}$$

Así, la tasa de salida R_{out} se puede expresar de la forma:

$$R_{out} = \frac{P}{T_{out}} = \frac{P}{T_{in}+\Delta q} = \frac{R_{in}C_t}{C_t+R_{in}-A} \quad \text{si } R_{in} > A \tag{2.13}$$

de manera que si $\Delta q = 0$ ==> $R_{out} = R_{in}$.

Este es el comportamiento principal de estas técnicas en las que utilizan el retardo de los paquetes y la diferencia de tiempos entre paquetes consecutivos para estimar el ABw.

Para comprender como funcionan las dos técnicas, se explicará en detalle en qué consisten los dos modelos PGM y PRT.

Probe Gap Model (PGM) [SKK03]

PGM es un modelo que se basa en la utilización de la información del tiempo entre la llegada de dos paquetes sucesivos en el receptor. Se envía un par de paquetes con un tiempo conocido entre ellos y suponiendo que el cuello de botella es único y que el tráfico cruzado es fluido, se observa esa diferencia de tiempo a la salida (véase Figura 5) y pudiéndose con ello calcular el ABw:

$$A = C(1 - \frac{\Delta_{out} - \Delta_{in}}{\Delta_{\text{in}}}) \tag{2.14}$$

donde $\Delta_{out} - \Delta_{in}$ es el tiempo que tarda en transmitirse el tráfico cruzado siendo $C(\frac{\Delta_{out} - \Delta_{in}}{\Delta_{\text{in}}})$ la tasa de dicho tráfico.

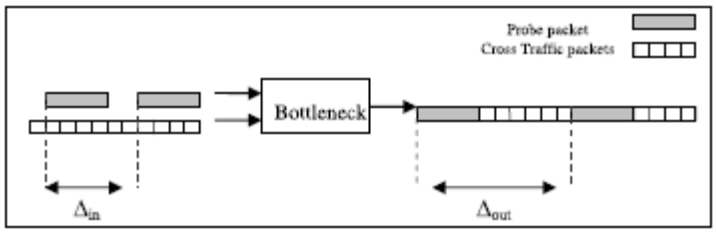

Figura 5. Modelo PGM para la estimación del ABw [SKK03].

Probe Rate Model (PRM) [SKK03]

PRM se basa en el concepto de congestión auto-inducida, de manera informal, si se envía el tráfico a un ritmo menor que el ancho de banda disponible, la tasa de llegada del tráfico en el receptor coincidirá con su tasa en el emisor. Por el contrario, si el tráfico se envía a un ritmo mayor que el ancho de banda disponible, los paquetes se acumulan dentro de la red y el tráfico se retrasa. Como resultado, la tasa en el receptor será menor que su tasa de envío. Así, se puede medir el ancho de banda disponible mediante la búsqueda de un punto de inflexión donde se dispare la relación entre las dos tasas.

2.3.1 Direct Probing Techniques (DP)

Esta técnica está basada, como ya se ha mencionado, en el modelo PGM, que utiliza la información del tiempo entre la llegada de dos paquetes sucesivos en el receptor.

La principal característica de esta técnica es que en cada flujo de medida se obtiene una estimación del ABw en tiempo real. El problema principal es que se necesita saber la capacidad total del enlace que se quiere medir para obtener dicha estimación. En los modelos PGM, la tasa de envió es única y corresponde a la capacidad del enlace a medir. De esta forma, acorde con 2.14, el ABw se puede obtener como sigue:

$$A = C_t - R_{in}\left(\frac{C_t}{R_{out}} - 1\right) \quad (2.15)$$

Como se puede ver, para emplear esta técnica, hay que conocer la capacidad del enlace, puesto que lo que hacen es calcular el ABw de forma directa (de ahí su nombre); Así, mediante el envío de unos pocos paquetes a una tasa contante, se obtiene una medida de la tasa de salida (R_{out}) o del tráfico cruzado y se le resta la proporción utilizada (R_{in}/R_{out}) de dicha capacidad, que tiene que ser conocida.

La característica principal de los modelos PGM es la relación matemática existente entre las tasa de salida y la de entrada bajo un modelo de tráfico fluido. Herramientas que usen este modelo son *Delphi* [Rib+00] o *Spruce* [SKK03], que se basan en la estimación del tráfico cruzado (*Rx*) a partir de un estudio matemático del retardo que sufren los paquetes al atravesar un enlace. Combinando estos algoritmos junto con la expresión 2.9 se puede estimar el ABw.

Estas técnicas son inadecuadas si no se conocen las capacidades de los enlaces de la red en los que se quiere realizar la medida. No obstante, existen diversas herramientas como *CapProbe* [Kap+04] que son capaces de estimar la capacidad de un enlace de modo que si se combinan las dos técnicas se puede estimar el ABw [CV06, LDS06].

2.3.2 Iterative Probing Techniques (ItP)

Las técnicas ItP son las que se han estudiado para este proyecto, por eso, en este apartado, dichas técnicas se van a explicar un poco más en detalle con el fin de introducirlas y dar un visión general de su funcionamiento. Posteriormente, en la siguiente sección, se desarrollarán de una forma más completa.

Estas técnicas siguen el modelo PRM, en las que se envían ráfagas a diferentes tasas en el origen y, después de atravesar la red, el receptor observa y analiza la tasa de salida y el retardo de los paquetes para determinar el ABw. Cuando la tasa de salida es menor que la de entrada o el retardo relativo de los paquetes se ve afectado, esto indica que la tasa utilizada (R_{in}) es mayor que el ABw. De esta forma, analizando los resultados de dicha tasa se obtiene finalmente el ABw. Estas técnicas son más interesantes debido a que no es necesario conocer las capacidades de los enlaces en la ruta a medir [LDS06, CV06].

Existen varias ItP pero este estudio se ha centrado en tres de ellas llamadas *Train of Packet Pairs* (TOPP), *Self-Loading Periodic Streams* (SLoPS) y *PathChirp.*

Train of Packets Pairs (TOPP) [CV06, **MBG00]**

Este método está basado en Packet Pairs Technique (PP), que consiste (ver Figura 6) en enviar flujos de pares de paquetes incrementando uniformemente la tasa de envío en cada iteración. La tasa se cambia variando el tiempo (T_{in}) entre los paquetes de cada par. Los paquetes tienen siempre el mismo tamaño P y utilizando la ecuación 2.11 se puede calcular la tasa de envío. Realizando medias del tiempo entre paquetes en el destino, se puede comprobar si la tasa de envío de origen supera el ABw del enlace.

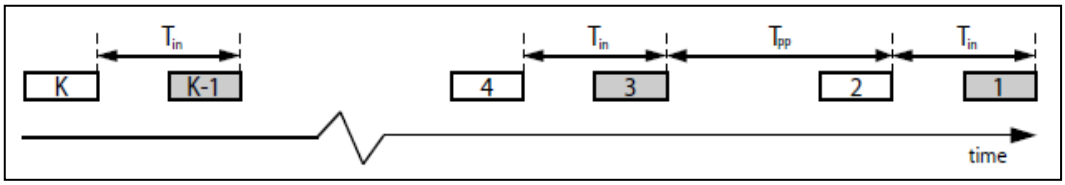

Figura 6. Esquema general de TOPP [CV06]

Uno de los problemas de este método es que al estar basado en PP es muy dependiente del tipo de tráfico cruzado que exista en el enlace. Para mitigar este efecto, es conveniente que el tamaño de los paquetes sea lo suficientemente grande.

Esta técnica está implementada en una herramienta llamada *DietToPP* [JMB04] en la que se simplifica un poco su implementación. En el presente estudio se ha simplificado un poco más el desarrollo y además se ha implementado en Java junto con otras herramientas para poder comparar.

Self-Loading Periodic Streams (SLoPS) [CV06,JD03]

Este es otro excelente ejemplo de ItP. Consiste en el envío de flujos de paquetes equidistantes en el tiempo. No está basado en PP pero al igual que en TOPP, la tasa de envío se incrementa en cada iteración, pero ésta vez, se realiza modificando el tamaño de los paquetes en vez de el tiempo entre ellos. Además, se utiliza como medida un análisis estadístico de la variación del retardo de los paquetes, es decir del OWD y no sobre tiempo entre cada paquete. También cambia la manera de incrementar la tasa de envío, en lugar de ser lineal como en TOPP, es binaria, de forma que la búsqueda del ABw es más eficiente.

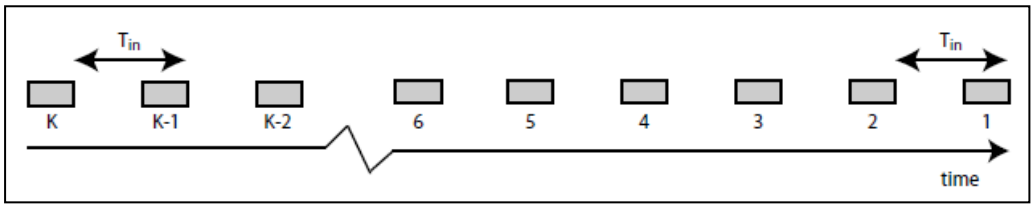

Figura 7. Esquema general de SLoPS [CV06]

Esta técnica se ha implementado en una herramienta llamada *Pathload* [JD03].

PathChirp [Rib+03,CV06]

Este es un método bastante preciso y es más eficiente que los anteriores. Su funcionamiento es parecido a TOPP, es decir, mantiene el tamaño de los paquetes (P) constante y aumenta el tiempo entre paquetes. La principal diferencia es que el aumento del tiempo entre paquetes es exponencial (véase Figura 8), de esta forma, la tasa de envío entre cada paquete K es:

$$R_k = \frac{P}{T_k} = \frac{P}{\delta \gamma^k} \quad k = 0, \ldots\ldots, K-2 \tag{2.16}$$

donde T_k es el tiempo instantáneo entre paquetes y δ es el tiempo entre los dos paquetes más cercanos del flujo. Así, se puede probar un rango de tasas en un mismo flujo, con lo

que no es necesario enviar varias iteraciones. Con una sola iteración es suficiente, puesto que en un flujo ya estarían probadas varias tasas.

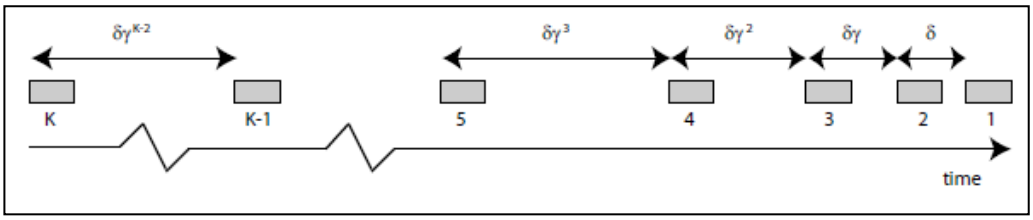

Figura 8. Esquema general de PathChirp [CV06]

Además, al igual que el SLoPS, también se utiliza el OWD, con el que, asumiendo un modelo de tráfico cruzado fluido, se realiza un estudio matemático para obtener el ABw. Con el fin de obtener más precisión en la medida se envían varios flujos y se obtiene una media de todos ellos para realizar la estimación final del ABw.

PathChirp no pertenece a las DP puesto que, aunque solo necesite una iteración, no necesita saber la capacidad del enlace y está basado en la medida del retardo producido por la congestión del enlace, como ocurría en las ItP.

2.3.3 Técnicas Mixtas [CV06]

Estas técnicas son una combinación de las dos anteriores. Como se ha mencionado en la introducción del apartado, *IGI* [HS03] es una herramienta que usa un algoritmo iterativo para estimar el tráfico cruzado y necesita la capacidad del enlace a medir para estimar el ABw. El método se basa en el envío de paquetes equiespaciados en el tiempo y se mide el tiempo de separación entre ellos con el que salen de la red (véase Figura 9). Con esto se puede estudiar el comportamiento del tráfico cruzado puesto que no todos los paquetes sufrirán el mismo retardo. Tan solo los paquetes que sufran algún retardo son los que se utilizarán para estimar el tráfico cruzado. La cantidad de tráfico cruzado durante el tiempo de prueba T_p es calculado por medio de la siguiente ecuación:

$$X = C_t \sum^{k^+} (T_{out}^+ - \frac{P}{C_t}) \qquad (2.17)$$

donde k^+ es el número de paquetes que han sufrido un incremento de tiempo.

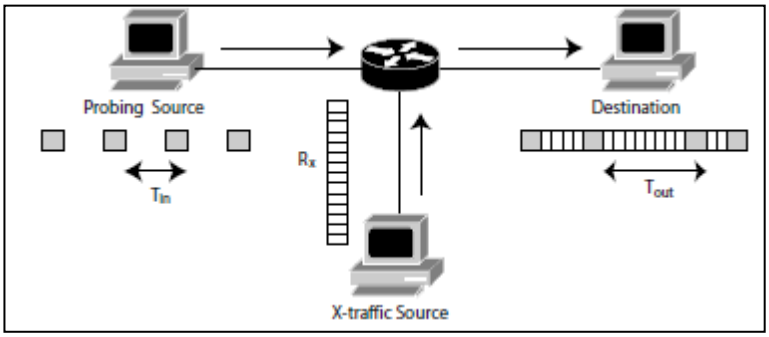

Figura 9. IGI [CV06]

El tráfico cruzado se puede calcular por medio de la siguiente expresión:

$$R_x = \frac{X}{T_p} = C_t \frac{\sum^{k^+}(T^+_{out} - \frac{P}{C_t})}{\sum^{k^+} T^+_{out} + \sum^{k^=} T^=_{out} + \sum^{k^-} T^-_{out}} \quad (2.18)$$

donde $k^=$ es la relación de paquetes que mantienen constante la separación entre sí, y k^- la relación de paquetes que disminuye dicha separación.

La clave del método consiste en encontrar la diferencia inicial idónea. Una diferencia de tiempo muy pequeña significa una gran tasa de entrada, que puede sobrecargar la red y no sería útil para estimar el tráfico cruzado. El método utiliza un algoritmo iterativo, que consiste en aumentar la tasa inicial, hasta que diferencia entre la suma de entrada y salida de los tiempos del tren sea tan pequeño como la deseada. En ese punto, la ecuación 2.18 se aplica para calcular el tráfico cruzado y el ABw se obtiene utilizando la ecuación 2.9.

El mayor problema radica en que se necesita saber la capacidad del enlace, pero el método muestra de forma intuitiva los efectos del tráfico cruzado.

2.4 Conclusiones

A continuación se resumen las técnicas y los métodos que se han introducido en el apartado anterior para tener una idea general de su funcionamiento:

- *Direct Probing Techniques (DP),* obtiene una estimación del ABw en cada flujo enviado midiendo la tasa de salida o estimando el tráfico cruzado. El ejemplo más claro es una herramienta llamada *Delphi* y que lo que calcula es el tráfico cruzado. La principal ventaja es su tremenda adaptación en tiempo real pero, por contra, es necesario saber la capacidad del enlace a medir.

- *Iterative Probing Techniques (ItP),* estima si la tasa de entrada es mayor o menor que el ABw en cada iteración. No se requiere el conocimiento de la capacidad del enlace a medir. Los algoritmos propuestos y descritos anteriormente utilizan este tipo de técnicas que se pueden resumir en:

    - *TOPP*: se envían pares de paquetes incrementando uniformemente la tasa de entrada en cada iteración. La tasa de envío se cambia modificando el tiempo entre paquetes. Se mide la variación del tiempo entre pares de paquetes que llega al receptor. Este método también se puede utilizar para estimar la capacidad del enlace más pequeño y el ABw de enlaces secundarios.

    - *SLoPS*: se envían paquetes separados una distancia constante. La tasa de envío se modifica variando el tamaño del paquete. Además mide el OWD que sufren los paquetes y se realiza una búsqueda binaria, lo que hace que sea más eficiente.

- *PathChirp*: en este caso se envían flujos de paquetes de manera que el espaciado entre paquetes va aumentando de forma exponencial. De esta forma la tasa de envío instantánea es múltiple en cada flujo y por ello sólo es necesaria una iteración. Como se podrá observar, aquí no se emplean pares de paquetes.

- *IGI*: este método utiliza una mezcla de técnicas. Está basado en enviar trenes de de paquetes incrementando el espaciado entre paquetes hasta que se iguala el espaciado de la entrada con el de la salida. Con ello puede medir el tráfico cruzado y con él, el ABw. La única pega es que, al ser una mezcla de DP e ItP, la capacidad del enlace medido tiene que ser conocida, pero este método es bastante eficiente y preciso.

Es importante destacar que el desarrollo de estos algoritmos ha buscado que la forma de medir el ABw sea lo más eficiente posible, es decir que no sea intrusiva para no afectar al tráfico que está cursando por la red y que además sea lo suficientemente precisa para dar una estimación aceptable.

En base a algunas pruebas realizadas [Cas+06], la eficiencia de los tres algoritmos expuestos puede basarse en el nivel de interferencia con la red, es decir, la carga y su velocidad. De esta manera podemos resumir esto en la Tabla 1:

Parámetro	TOPP	SLoPS	PathChirp
Carga (KB)	977	1079	127
Tiempo de Prueba (s)	19.96	3.96	0.76
Error Medio (Mbps)	0.40	2.23	1.37

Tabla 1. Comparación de la eficiencia de los algoritmos [Cas+06]

Podemos observar como PathChirp es el más eficiente en cuánto a carga y tiempo, puesto que necesita muy pocos KBytes y poco tiempo para realizar una medida. También hay que mencionar, que TOPP es el menos intrusivo pero debido a que utiliza más tiempo de prueba (más iteraciones), en cuanto a carga es bastante más elevado que PathChirp. SLoPS se colocaría entre los dos algoritmos que con capacidades similares a las de TOPP, necesita menos tiempo de prueba pero necesita muchos paquetes.

Hay que destacar dos consideraciones que se han tenido en cuenta a lo largo del estudio de estas técnicas cuando se prueban en una red real [CV06]:

- Se considera que los routers encolan paquetes con una política FIFO, lo cual no tiene porqué ser cierto en toda la red, aunque es lo más probable.

- También se asume que el camino por donde van todos los paquetes es el mismo, es decir, que los paquetes llegan ordenados y pasan por los mismos enlaces.

- Se asume un modelo de tráfico cruzado constante, lo que no tiene porqué cumplirse a lo largo de la parte donde está el cuello de botella, que es la parte que interesa, pero en intervalos pequeños de tiempos de medida, esta parte se puede considerar cierta.

Capítulo 3

Estimación del ABw

En este capítulo, se van a detallar las técnicas desarrolladas y empleadas en el proyecto. Por ello sólo se explicarán las ItP, que son las técnicas en las que se basan los tres algoritmos que se han implementado en la herramienta. Con este capítulo se da por terminado el estudio teórico de los algoritmos y posteriormente se pasará a explicar el desarrollo de la herramienta y la implementación de los algoritmos en dicha herramienta.

3.1 Iterative Probing Techniques (ItP)

Primero se definirán algunos conceptos fundamentales sobre esta tecnología. Como ya se ha mencionado, la principal ventaja de estos algoritmos es que el proceso de medida casi no afecta al ABw real y no sobrecarga la red a la vez que se realiza dicho proceso.

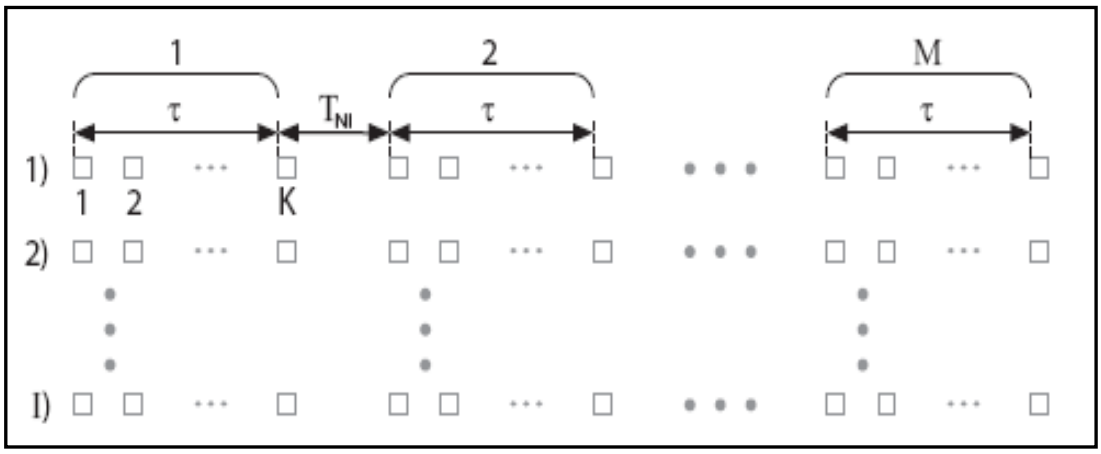

Figura 10. Estructura general del envío de paquetes mediante ItP [Cas+06]

Esta técnica está basada principalmente en el envío de un número de trenes o flujos (M) de paquetes, tal y como se muestra en la Figura 10. Cada tren contiene K paquetes, espaciados en un tiempo determinado. La tasa con la que se van enviando dichos trenes de paquetes se va cambiando en cada iteración en función de si se han recibido dichos paquetes en el receptor. La máxima tasa que haga que los paquetes lleguen sin sobrecarga será el ABw en el enlace.

En la Tabla 2. Parámetros principales de ItP definimos los parámetros comunes de este tipo de técnicas:

Parámetro	Definición
τ	Time-scale, es el tiempo en el que el flujo interactúa con el tráfico cruzado en un determinado momento, es decir la duración de cada flujo.
T_{NI}	Tiempo de no intrusión, es el tiempo durante el cual no se emite nada, para evitar conflictos con el siguiente flujo y poder evaluar la tasa con mayor precisión.
I	Es el número de iteraciones..
M	Es el número de flujos que se envían por cada iteración. Para mejorar la precisión de la medida se realizan varios flujos y se calcula su media
K	Es el número de paquetes de longitud L que se envían por cada flujo M.

Tabla 2. Parámetros principales de ItP

3.2 Análisis de las técnicas implementadas

En este apartado se realiza una estudio exhaustivo de los métodos que se han introducido y además, son los que se han desarrollados en la herramienta. Se explicarán los tres algoritmos descritos, TOPP, SLoPS y PathChirp. Se estudiarán cada una de sus características, los parámetros y las forma en que cada uno ellos obtienen la medida final del ABw.

3.2.1 Train Of Packet Pairs (TOPP)

En este apartado se explicará el funcionamiento del algoritmo llamado TOPP de forma que primero se describirán sus parámetros y su fase de análisis y después se explicarán una serie de consideraciones que hay que tener en cuenta a la hora de utilizarlo.

TOPP es un algoritmo basado en la técnica de Pares de Paquetes (PP). Esta técnica está basada, como ya se ha mencionado anteriormente, en transmitir varios trenes de pares de paquetes. En cada iteración se prueba una tasa (R) diferente. Dicha tasa se modifica reduciendo el espaciado entre los pares de paquetes, manteniendo siempre el tamaño del paquete (L) constante.

3.2.1.1 Descripción del Algoritmo TOPP[CV06]

En la Figura 11, se puede observar un esquema del funcionamiento de TOPP. En cada iteración se va disminuyendo el tiempo entre parejas de paquetes (T_{in}^{n}), de forma que se va aumentando el tiempo entre pares de paquetes (T_{PP}^{n}). Así se mantiene constante el número de pares de paquetes.

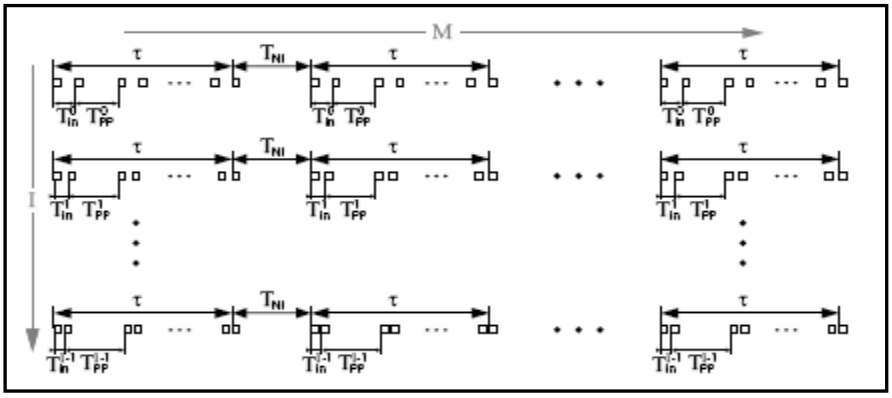

Figura 11. Esquema TOPP [CV06]

El envío de dichos flujos de paquetes se realiza M veces por iteración con el fin de obtener una mayor precisión en la medida.

Parámetros y expresiones relevantes

TOPP tiene dos parámetros que se van actualizando en cada iteración que se realiza, de forma que cada vez, se envían pares de paquetes a una tasa constante R_{in}. La diferencia de tiempos entre los dos paquetes consecutivos (T_{in}) y entre los pares de paquetes (T_{pp}), es la que determina la tasa de envío. Esta tasa se modifica en función del parámetro *w* llamado *factor de resolución*. De esta forma, en cada iteración *i*, la tasa de envío se va modificando de acuerdo con la expresión 3.1,

$$R_{in}^{i} = \mathrm{R}_{\min} + \ i * w \tag{3.1}$$

El número de iteraciones I se puede obtener de la forma:

$$I = \frac{R_{max} - R_{min}}{w} + 1 \tag{3.2}$$

TOPP envía N=K/2 pares de paquetes en cada flujo. De esta manera, el cálculo de τ se realiza de la siguiente manera:

$$\tau = T_{in}\frac{K}{2} + T_{pp}\left(\frac{K}{2} - 1\right) \quad (3.3)$$

Es importante que el tiempo de prueba esté acotado entre valores razonables para poder obtener una medida los más precisa posible, puesto que la medida se tiene que realizar durante condiciones de tráfico lo más constante posible para poder obtener el ABw de forma más efectiva posible. Por otro lado, como el tamaño del paquete siempre es constante (L), el parámetro que va a determinar la tasa de entrada es T_{in} y T_{pp} es el valor que se ajusta para poder mantener constate τ.

$$T_{PP}^{i} = \frac{\tau - T_{in}^{i}\frac{K}{2}}{\frac{K}{2}-1} \quad i = 0, \ldots, I-1 \quad (3.4)$$

donde I es el número total de iteraciones.

T_{PP}^{min} se debe escoger de forma que no interfiera en tiempo en la misma cola que los pares anteriores y siguientes. Sin embargo, es complejo determinar de forma óptima un valor de T_{PP}^{i} de forma independiente al tipo de tráfico de la red. En la práctica, se establece un porcentaje η sobre el tiempo máximo entre paquetes:

$$T_{PP}^{min} = (1 + \eta)T_{in}^{max} \quad (3.6)$$

donde $\eta > 0$ para mantener la tasa entre los dos pares de paquetes, fijado por T_{pp}, por debajo de la tasa de entrada en ese momento. Para un τ y una R_{min} dados, η debe de ser lo más grande posible para evitar la influencia del paquete sobre el siguiente pero se debe llegar a un compromiso, puesto que al incrementar dicho parámetro, el número de paquetes de cada flujo disminuye.

La medida del ABw se realiza de forma que enviando paquetes a una tasa constante, es decir con T_{in} constante, si al atravesar el enlace su valor permanece constante o no se aleja demasiado del valor inicial, se puede estimar que el ABw es el la tasa con la que se están enviando los pares de paquetes:

$$R_{in} = \frac{L}{T_{in}} \quad R_{out} = \frac{L}{T_{out}};\ Tout \simeq Tin \Rightarrow Rin \simeq Rout \quad (3.5)$$

donde, R_{in} es la tasa de envío inicial, R_{out} es la tasa obtenida a la salida y T_{out} es el tiempo medido entre pares de paquetes a la salida del enlace.

En la Figura 12 se muestra como es la dispersión producida en los paquetes cuando atraviesan el *cuello de botella*.

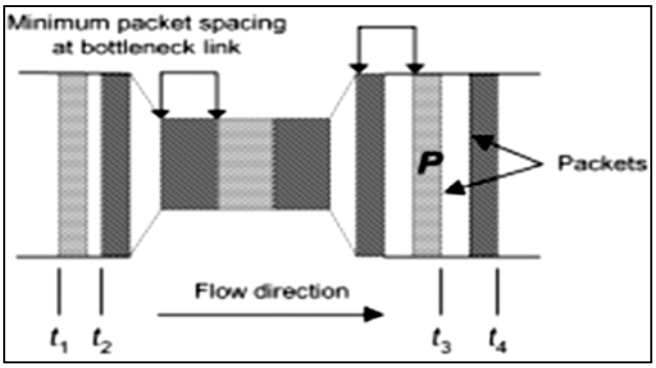

Figura 12. Dispersión de paquetes [CM02]

Se puede ver como los paquetes, a la salida del *cuello de botella*, están separados un tiempo mayor que cuando entraron. Esta dispersión es la que se mide con este algoritmo (T_{out}). De esta manera, se puede comprobar si la tasa de envío sufre alguna variación ,es decir, si los paquetes se separan o se juntan a su salida.

Es importante que el tamaño de los paquetes esté entre 200 y 1500 bytes (cabeceras incluidas), para que cumplan con los requisitos de la MTU y para que no sean fragmentadas por IP. Además, debe ser múltiplo de 92, para evitar el relleno de ceros (padding) que se auto genera en los enlaces ALL5-ATM y por el mismo motivo existe un tamaño mínimo para la trama Ethernet. Esto será una condición que se aplicará a todos los algoritmos por igual.

También se introduce un T_{NI} que es el tiempo suficiente para que no se mezclen las muestras enviadas entre sí y así obtener unos resultados lo más independientes posibles.

Además de estos parámetros, se puede establecer una relación para mantener constante el tiempo de medida y la tasa mínima y así poder reducir el número de variables:

$$\tau R_{min} = L[(2+\eta)N - (1+\eta)] \tag{3.7}$$

de esta forma queda determinado τ en función de varios parámetros. Por otro lado, la tasa máxima queda determinada por tiempo T_{in} mínimo que deje el sistema operativo y el hardware utilizado.

Proceso de medida y análisis de los resultados

Después de haber recibido los paquetes enviados a una tasa constante, se analizan los tiempos de llegada y se calcula la relación entre la tasa de entrada y la de salida usando la expresión 3.8. Un vez realizado esto, se envía la siguiente iteración con la siguiente tasa y así sucesivamente hasta que se llegue a la tasa máxima establecida (R_{max}).

$$R = \frac{R_{in}}{R_{out}} \tag{3.8}$$

Una vez obtenidas todas las medidas, el cliente calcula el ABw realizando una análisis de la relación entre la tasa enviada y recibida, esto es, si $R \approx 1$ quiere decir que la tasa enviada no tiene sobrecarga, con lo que se puede probar la siguiente. En el momento en el que $R >> 1$ se indica que empieza a existir sobrecarga y ya no se dispone de ese ABw.

Esto se muestra en la Figura 13, donde se puede apreciar que en la primera parte, hasta 10 Mbps, la relación R se aproxima a 1 y luego se dispara, con lo que el ABw ≈ 10 Mbps.

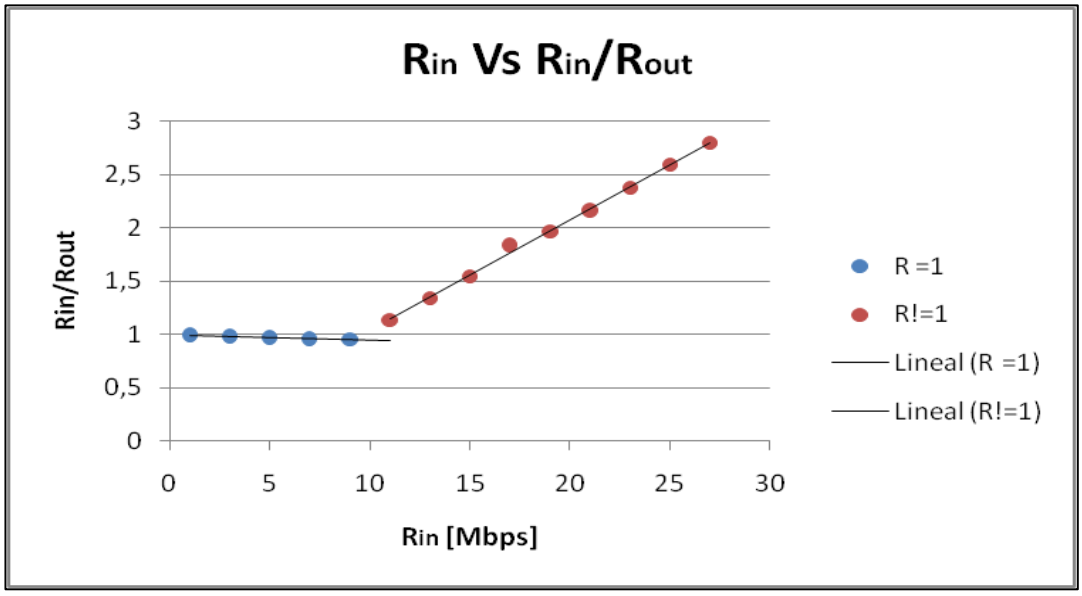

Figura 13. Relación Rin/Rout en un test TOPP

Existen diversas formas para obtener ese punto de inflexión. Uno de ellos es el método de la máxima varianza. Una vez obtenida la nube de puntos, se calcula las rectas a las que se aproximan los puntos. Después, para ver en qué punto la nube cambia la pendiente, se calcula varianza σ^2 sobre cada punto de la nube. El punto que tenga menos varianza, es el punto de inflexión.

Otro método, es el de la segunda derivada. Consiste en calcular la función "y" de la nube de puntos y obtener el punto dónde su segunda derivada es mayor. Este método, es más sensible al ruido que pueda existir en la nube de puntos comparado con el método de la mínima varianza, pero es un poco más fácil de implementar.

En la Figura 14 se refleja una muestra de los dos ejemplos mencionados.

Debido a la complejidad de estos métodos, se ha decidido implementar la solución de una forma más sencilla que se desarrollará posteriormente, en el apartado sobre la implementación del algoritmo en la herramienta.

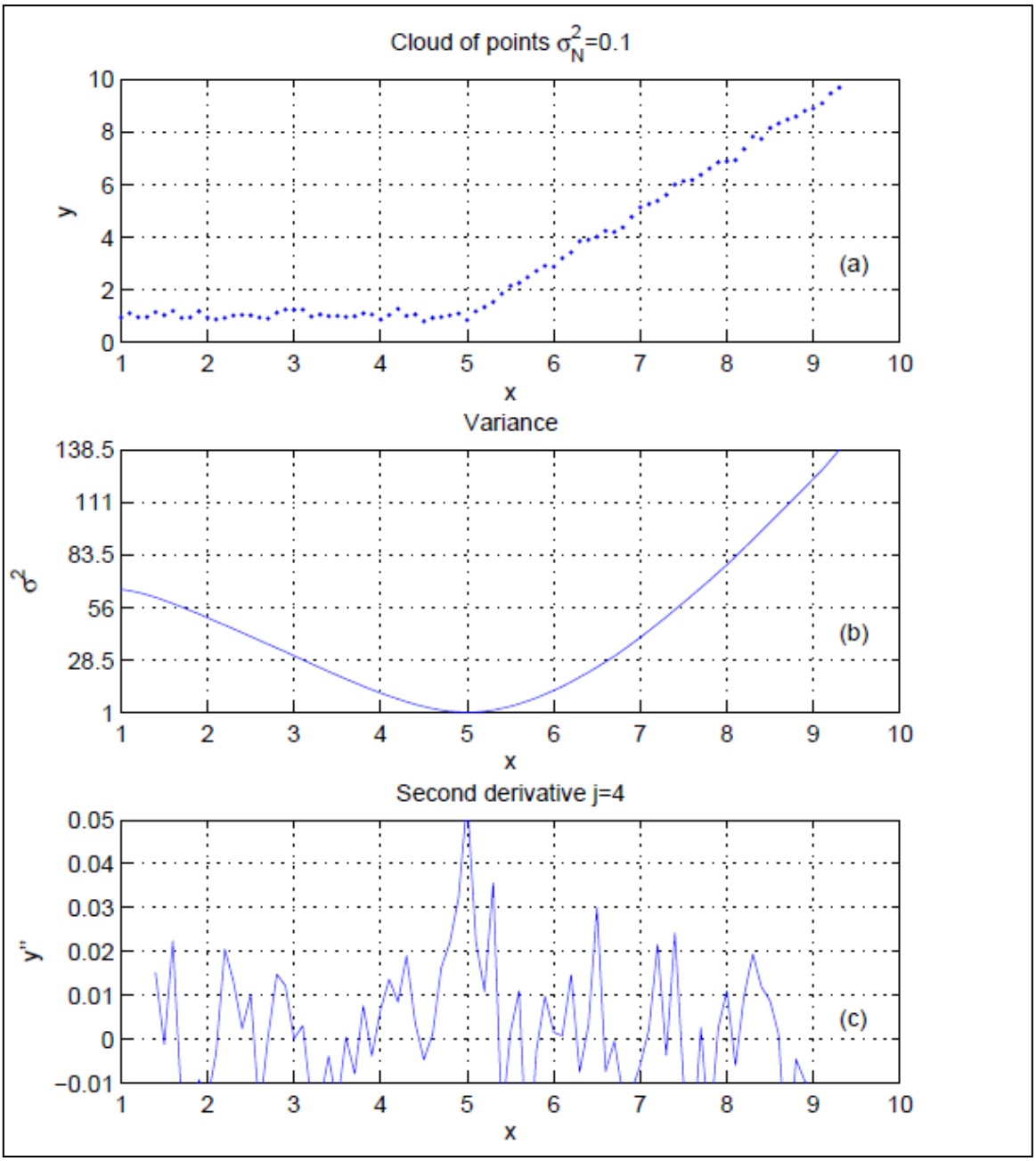

Figura 14. Cálculo del punto de inflexión en TOPP, b) Mínima varianza c) Segunda derivada [CV06]

3.2.1.2 Consideraciones en TOPP [CV06]

TOPP tiene una serie de consideraciones que hay que tener en cuenta para que las medidas realizadas sean lo más reales posibles y así su rendimiento sea eficiente.

Resolución frente a tiempo de prueba

De la ecuación 3.2 se deduce que el número de iteraciones es inversamente proporcional a la resolución deseada. Con esto se quiere decir que, cuanto mayor sea la resolución (más cercana a cero) mayor será el número de iteraciones necesarias, y su consecuencia será que mayor será el tiempo de medida. Esto puede suponer un problema puesto que

dependiendo del tipo de red donde se esté realizando la medida, si el tráfico es muy cambiante en poco tiempo, la medida será menos efectiva, puesto que si se tarda demasiado en medir el algoritmo no funcionará correctamente.

Efecto borde

Puede que el algoritmo no detecte de forma muy precisa el punto de inflexión debido al posible ruido que se tenga al realizar la medida. Para poder resolver este problema es posible que se tenga que aumentar el número de iteraciones y aumentar la resolución.

Tamaño del paquete del tráfico cruzado

Este algoritmo, al estar basado en una técnica Packet Pair (PP), es muy sensible al tipo de tráfico cruzado que exista en la red que atraviesa. Por ello, es necesario tener una idea del tamaño que tienen los paquetes del tráfico existente de la red. Según algunos estudios realizados en una red que atraviesa Internet, si el tamaño L de los paquetes utilizados para realizar la medida es mayor a 800 bytes [DRM01], el algoritmo obtiene mejores resultados. Esto es debido a que, al utilizar paquetes con un tamaño similar a la media que se utiliza en Internet, el algoritmo se hace menos sensible a este parámetro pero por contra, se aumenta el tiempo de prueba.

3.2.2 Self-Loading Periodic Streams (SLoPS)

Self-Loading Periodic Stream es un algoritmo similar a TOPP. En este caso, lo que se modifica para variar la tasa de envío, es el tamaño del paquete y se mantiene constante la separación entre los paquetes.

SLoPS está basada en la auto-congestión, es decir, la propia ráfaga de paquetes enviada, en función de la tasa de envío, es la que provoca una pequeña carga en la red que se puede detectar midiendo el OWD. De esta forma, realizando un pequeño análisis del OWD se puede saber si la tasa de envío es mayor o menor que el ABw.

Este algoritmo se ha implementado en una herramienta llamada *pathload* [JD03], con la que se han realizado estudios similares al descrito en este proyecto.

3.2.2.1 Descripción del Algoritmo SLoPS[CV06]

En la Figura 15 se representa un esquema de la forma de envío en cada iteración. En dicha figura se puede ver el resultado de varias iteraciones. Vemos que en cada iteración, todos los paquetes tienen el mismo tamaño (L) y están separados el mismo tiempo T_{in}. En la siguiente iteración, el tamaño de los paquetes (L) se incrementará o disminuirá para poder variar la tasa de envío.

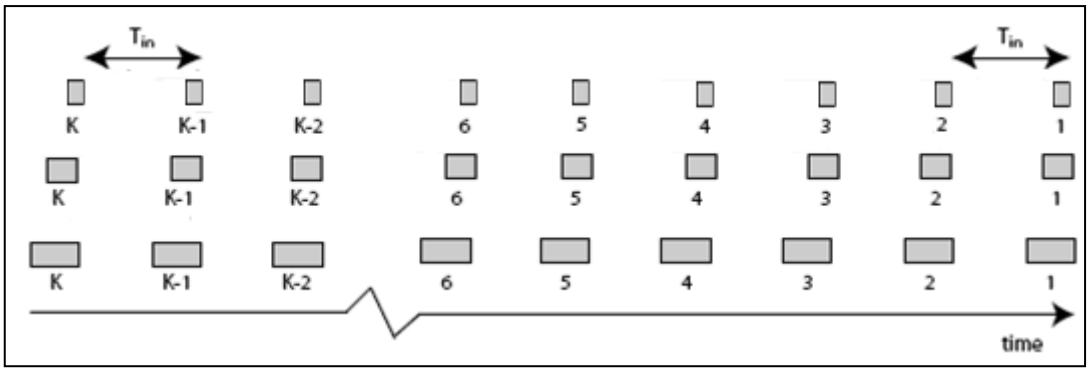

Figura 15. Esquema SLoPS

A diferencia del TOPP, SLoPS no es un algoritmo basado en PP con lo que no es tan sensible al tráfico cruzado. Además, utiliza un sistema binario en lugar de uno lineal, para encontrar el ABw, lo que reduce considerablemente el número de iteraciones necesarias. Al igual que en TOPP, se enviarán varios flujos (M) para poder obtener una mejor medida del ABw.

En cada flujo se medirá el OWD y se detectará si ha habido algún incremento de tiempos entre paquetes consecutivos, es decir, un incremento en las variaciones de los OWDs. De esta manera se podrán distinguir los trenes de paquetes que han sufrido incremento y los que no, y en base a ello se realizará el análisis para obtener la siguiente tasa a probar.

Parámetros y expresiones relevantes

Al igual que en TOPP, T_{in} se calcula con la expresión 3.5, y con ello, a través de la expresión 3.9 con la selección de L y R_{in}, y después se puede calcular τ, que es el tiempo que va durar cada flujo [CV06]:

$$\tau = T_{in}(\mathrm{K}-1) \tag{3.9}$$

Es muy importante que el tiempo T_{in} sea lo más pequeño posible para que τ también lo sea y así se conseguirá que la medida sea menos intrusiva en el tiempo. Esto dependerá del hardware y software que se esté utilizando para realizar las medidas.

Al igual que en TOPP, también se introduce un T_{NI} que es el tiempo suficiente para que no se mezclen las muestras enviadas entre sí y así obtener unos resultados lo más independientes posibles.

En la misma línea, en este algoritmo, las tasas máxima y mínima (R_{min} y R_{max}) quedan determinadas por el tamaño máximo y mínimo (L_{min} y L_{max}) de los paquetes, que es el mismo marcado en TOPP, entre 200 y 1500 bytes con cabeceras [JC02] y recordando que debe ser múltiplo de 92, para evitar el relleno de ceros (padding) que se auto genera en los enlaces ALL5-ATM.

Como se ha mencionado antes, en este algoritmo se utiliza el OWD para medir el ABw. En realidad, lo que se utiliza es el OWD relativo entre cada paquete, es decir, la diferencia entre el OWD de cada paquete:

$$D_i = \mathrm{t_{ri}} - \mathrm{t_{si}} \quad i = 1,2,..K \tag{3.10}$$

donde t_r es la marca de tiempo tomada en el receptor y t_s una marca de tiempo tomada en el emisor. De esta forma, podría entenderse que el emisor y el receptor tienen que estar sincronizados pero como lo que se utiliza realmente son las variaciones entre las diferencias de los OWDs, no es necesario puesto que el offset que exista entre el emisor y el receptor es siempre el mismo y se elimina:

$$D_{i+1} + \mathrm{Offset_i} - (\mathrm{D_i} + \mathrm{Offsett_i}) \quad i = 1,2,..K-1 \tag{3.11}$$

En la expresión 3.11 podemos ver que si los dos offset son iguales se eliminan quedando solo la diferencia de OWDs, y lo que se utiliza es la variación del OWD entre pares consecutivos:

$$\Omega_i = D_{i+1} - \mathrm{D_i} \quad i = 1,2,..K-1 \tag{3.12}$$

Una vez obtenidas las variaciones del OWD entre cada par de paquetes se comprobará si el tren de paquetes es creciente o decreciente. Si la variación del OWD ha sufrido un incremento, es un tren creciente y viceversa.

La forma de estimar si el tren es no creciente o creciente se realiza mediante el estudio estadístico de dos parámetros, *Pairwise Comparasion Test* (S_{PCT}), que mide la fracción consecutiva de variaciones del OWD que han sufrido un incremento, y la *Pairwise Different Test* (S_{PDT}) que mide la fuerza de la variación extremo a extremo. Estos dos parámetros se calculan de la siguiente manera [CV06]:

$$S_{PCT} = \frac{\sum_{l=2}^{\Lambda} \Phi(\widetilde{\Omega}_l > \widetilde{\Omega}_{l-1})}{\Lambda - 1} \tag{3.13}$$

$$S_{PDT} = \frac{\widetilde{\Omega}_\Lambda - \widetilde{\Omega}_1}{\sum_{l=2}^{\Lambda} |\widetilde{\Omega}_l - \widetilde{\Omega}_{l-1}|} \tag{3.14}$$

donde hay que aclarar varios puntos:

$\Phi(X) = 1$ si cumple la condición X y $\Phi(X) = 0$ si no la cumple.

$\widetilde{\Omega}$ es un promedio de los OWDs calculado dentro de un mismo flujo y se calcula de la siguiente manera:

En primer lugar, en cada flujo, los K OWDs $\{D_1, D_2, D_3, ..., D_K\}$ obtenidos, se divide en $\Lambda = \sqrt{K}$ grupos.
Después, se calculan las medias correspondientes a cada grupo, estos son los $\widetilde{\Omega}$ de cada grupo.

Finalmente, se aplican las dos expresiones 3.13 y 3.14.

Esta técnica de partición y toma de promedios hace que el proceso sea más robusto, eliminando valores atípicos en el proceso de análisis del OWD.

De esta forma, se puede ver que los valores de cada parámetro pueden ser:

S_{PCT}: $0 \leq S_{PCT} \leq 1$ con 1 indicando que ha habido una tendencia creciente y si las variaciones de los OWDs son independientes entre sí, el valor esperado debería valer $S_{PCT} = 0.5$. Los umbrales que indican si se ha producido una tendencia creciente o decreciente son los siguientes: $S_{PCT} > 0.66$ si se ha producido una tendencia creciente, y si se ha producido una tendencia no creciente $S_{PCT} < 0.54$. Si el resultado está entre esos valores el resultado es ambiguo.

S_{PDT}: $-1 \leq S_{PDT} \leq 1$ con 1 indicando que se ha producido una tendencia creciente y si las variaciones de los OWDs son independientes entre sí, el valor esperado debería ser nulo. Para este parámetro, los umbrales son los siguientes: $S_{PDT} > 0.55$ si se ha producido una tendencia creciente y $S_{PDT} < 0.45$, si se ha producido una tendencia no creciente. Si el resultado está entre esos valores el resultado también será ambiguo.

En la Tabla 4 se muestra un resumen de los valores de S_{PCT} y S_{PDT}.

Parámetro	Rango	Umbral Máximo	Umbral Mínimo
S_{PCT}	$0 \leq S_{PCT} \leq 1$	0.66	0.54
S_{PDT}	$-1 \leq S_{PDT} \leq 1$	0.55	0.45

Tabla 3. Valores S_{PCT} y S_{PDT} [JD03]

La decisión sobre si el flujo ha mostrado un comportamiento creciente o no, se toma en función de esos umbrales como una función AND, es decir, si los dos muestran un comportamiento creciente, el flujo es creciente. Si los dos muestran un comportamiento no creciente el flujo es no creciente, pero si cada uno de los dos son opuestos, el flujo es ambiguo y se descarta de la medida.

Estos umbrales se han utilizado en la herramienta ya mencionada *pathload* [JD03].

Proceso de media y análisis de los resultados

Una vez obtenido los valores de los parámetros S_{PCT} y S_{PDT} en cada uno de los flujos enviados, se calcula cuántos de ellos pertenecen a una tendencia creciente, no creciente o ambigua, y se define otro parámetro llamado *fracción* (f), con el que se toma la decisión de si se ha encontrado un tipo de tendencia u otro.

El proceso es como sigue:

Se calcula la fracción de los flujos que muestran una tendencia creciente y la fracción de los flujos que muestra una tendencia no creciente. Si la fracción con respecto al total de flujos que han mostrado una tendencia creciente es superior al parámetro *f* , entonces se considera que la iteración ha mostrado una tendencia creciente y esto quiere decir que ha existido un cierto retraso en los paquetes, *región creciente*, con lo que la tasa utilizada, R_{in} es mayor que el ABw.

Por otro lado, si la fracción es inferior a *f*, entonces muestra lo contrario, que no ha existido retraso, *región no creciente,* y la tasa utilizada R_{in} es menor que el ABw.

En estos dos casos, cuando $R_{in} > A$, entonces $R_{max} = R_{in}$ y en caso contrario, si $R_{in} < A$ entonces $R_{min} = R_{in}$. Y la condición de finalización es cuando $R_{max} - R_{min} \leq w$, donde *w* es el factor de resolución determinado por el usuario.

Puede ocurrir que ninguna de las dos supere el valor de *f*, en ese caso se marca como *zona o región gris*, donde no queda determinado si se ha experimentado una tendencia creciente o no. En esta zona se determinará si $R \approx ABW$ en función de otro parámetro llamado *factor de resolución de la región gris* (*X*). Si se detecta una *región gris*, se establecerán otros valores máximos para esta zona, que son G_{max} y G_{min}. Así, la condición de finalización quedaría completada con otras dos condiciones más. La condición final se muestra en la expresión 3.15 [JD03]:

$$(R_{max} - R_{min} \leq w \;\|\; (R_{max} - G_{max} \leq x \;\&\&\; G_{min} - R_{min} \leq X)) \qquad (3.15)$$

En la herramienta *pathload*, f = 70% y $X = 2w$.

En la Figura 16 se pueden ver algunos ejemplos gráficos de lo anteriormente descrito.

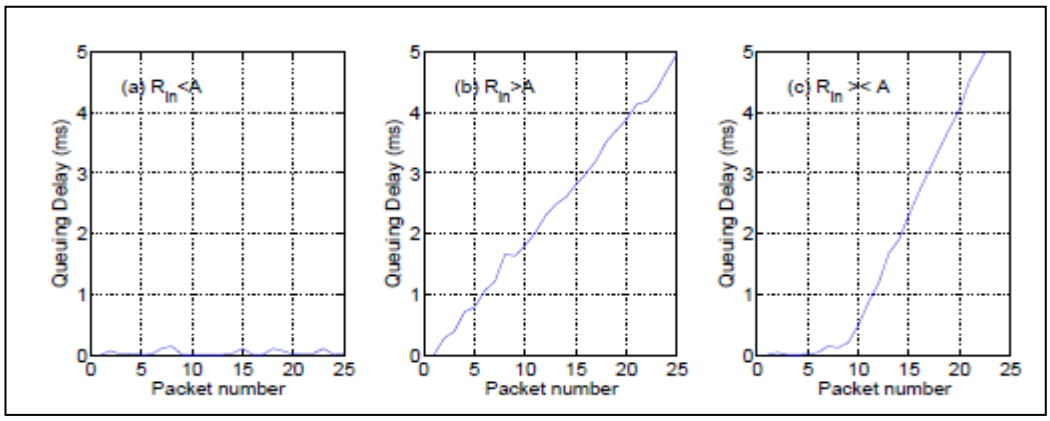

Figura 16. ΔOWD cuando a) R < ABw b) R > ABw c) R ≈ ABw [CV06]

Además, en la Figura 17 y Figura 18 se puede ver un ejemplo de una medida realizada con los parámetros S_{PCT} y S_{PDT}.

En el ejemplo de la Figura 17 y de la Figura 18 se puede deducir lo siguiente:

- En la primera iteración, (R = 94 Mbps, A = 75 Mbps), todos los flujos están en la *región creciente* tanto en el S_{PCT} como en el S_{PDT}, lo cual indica que R > ABw.

- En la segunda iteración, (R = 80 Mbps, A = 74 Mbps), se muestran 11 flujos en la *región creciente*, 6 en la *región no creciente*, y 3 han sido descartados. Esto indica que se está en la *región gris*, que determina que el R ≈ ABw.

- En la tercera y última iteración (R = 40 Mbps y A = 74 Mbps), todos los flujos se encuentran en la *región no creciente*, con lo que R < ABw.

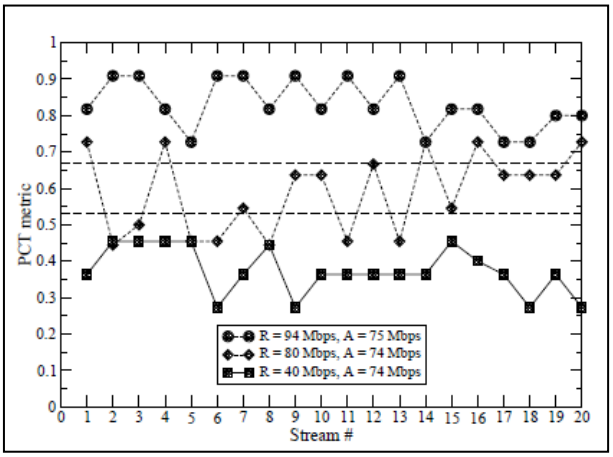

Figura 17. S_{PCT} 3 iteraciones (20 flujos) [JD03]

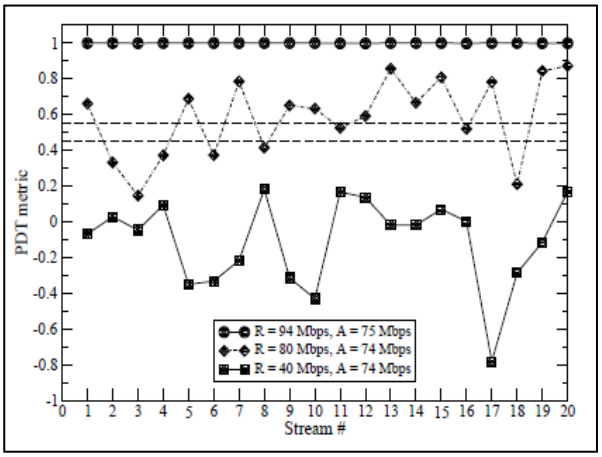

Figura 18. S_{PDT} 3 iteraciones (20 flujos) [JD03]

En el apéndice 3 se muestra en pseudo-código un ejemplo de cómo puede ser el algoritmo de búsqueda binaria descrito anteriormente.

3.2.2.2 Consideraciones en SLoPS [CV06]

Al igual que TOPP, en SLoPS hay que tener en cuenta una serie de consideraciones con las cuales se mejora su rendimiento.

Resolución frente a tiempo de prueba

Como ocurre en TOPP, la resolución es inversamente proporcional al número de iteraciones y como consecuencia de ello, al tiempo de prueba. En SLoPS, el número de iteraciones puede ser menor que en TOPP debido a su búsqueda binaria, pero también puede ser impredecible cuando se encuentran *regiones grises*. En caso de no encontrar dichas regiones, el número de iteraciones disminuiría considerablemente determinado por la expresión 3.16.

$$I = \geq log_2 \left(\frac{R_{max} - R_{min}}{w}\right) \tag{3.16}$$

Rango de la medida del ABw

Como ya se ha mencionado en la descripción del algoritmo, para evitar los efectos de *padding* en la capa Ethernet, el tamaño mínimo del paquete tiene que ser de 200 bytes. Y para evitar la fragmentación de IP, el tamaño máximo tiene que ser 1500 bytes (todo ello con cabeceras). De esta forma, la tasa máxima y mínima quedan limitadas por la expresión 3.17:

$$R_{min} = \frac{L_{min}}{T_{in}} \qquad R_{max} = \frac{L_{max}}{T_{in}} \tag{3.17}$$

con ello el rango queda determinado por el tiempo T_{in} mínimo que se pueda seleccionar.

3.2.3 PathChirp

En este apartado, se describe el último algoritmo estudiado, *PathChirp* [Rib+03]. *PathChirp* tiene como principal ventaja la poca intrusión que tiene en la red así como la rapidez con la que efectúa la medida. Esto se debe a que solo necesita una iteración para realizar la medida del ABw.

En este caso, el tamaño del paquete permanece constante y lo que se varía es el espacio en tiempo entre paquetes, pero esa variación, como ya se describirá, es exponencial y no lineal como se realiza en TOPP.

Al igual que en SLoPS, *PathChirp*, está basada en la auto-congestión y también mide el OWD.

Este algoritmo está implementado en una herramienta llamada como su propio nombre, *pathChirp* [Rib+03].

3.2.3.1 Descripción del Algoritmo PathChirp [CV06]

PathChirp es una herramienta que está diferenciada del resto debido a que en tan sólo una iteración puede dar una estimación del ABw. Esto se debe a que utiliza estructura exponencial en cada flujo. En la Figura 19 se puede observar dicha estructura:

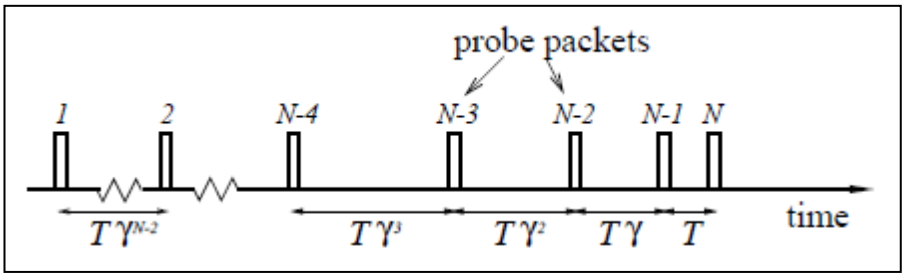

Figura 19. Estructura PathChirp [Rib+03]

PathChirp envía trenes de paquetes como los representados en la Figura 19. En este caso, sólo se realiza una iteración enviando M flujos para obtener una mejor estimación en la medida.

Además, en comparación con TOPP, se requieren N paquetes y N-1 medidas del espacio entre paquetes, mientras que en TOPP se necesitan 2N-2, con lo que de esta manera, se reduce considerablemente la cantidad de medidas necesarias.

Por otro lado, al igual que SLoPS, utiliza la variación de los OWDs para realizar la estimación. Realiza un estudio distinto al presentado en SLoPS que hace que el método sea mucho más rápido y eficaz.

PathChirp se considera una de las herramientas más eficientes basadas en *chirp*[3]. Así se puede obtener mucha información acerca del retardo y calcularlo rápidamente con un solo *chirp* puesto que se pueden medir varias tasas simultáneamente. Naturalmente, se envían más flujos para darle más robustez y consistencia a la medida.

Parámetros y expresiones relevantes

En *PathChirp*, los parámetros más inmediatos a calcular son T, δ y N (en nuestra nomenclatura es K) (véase Figura 19), puesto que son los parámetros que van a determinar la separación entre los K paquetes. Para ello se utilizan las expresiones mostradas en 3.18 [CV06]:

$$R_{min} = \frac{L}{\delta \gamma^{K-2}} \qquad R_{max} = \frac{L}{\delta} \tag{3.18}$$

donde δ hace el papel de T y corresponde al tiempo mínimo con el que se pueden separar los paquetes, es decir, el que determina la tasa máxima a medir. Por otro lado, γ es el factor de propagación, que es el que va incrementando el espacio entre los paquetes y así va disminuyendo la tasa de envío. Es importante ver que ese incremento se realiza de forma exponencial y además que $\gamma > 0$. En estudios realizados [Rib+03], el valor óptimo del factor de propagación es $\gamma = 1.2$. También se puede comprobar que el número de paquetes K enviados está limitado por este factor y por las tasas máxima y mínima de envío (R_{min}, R_{max}).

Una vez determinado esto, es fácil calcular el tiempo de duración del *chirp*, es decir τ, el cual se puede expresar como muestra en 3.19 [CV06]:

$$\tau = \sum_{k=0}^{K-2} T_k = \sum_{k=0}^{K-2} \delta \gamma^k = \frac{1-\gamma^{K-1}}{1-\gamma} \delta \tag{3.19}$$

Al igual que en los dos algoritmos anteriores, también se introduce un T_{NI} entre flujos y así se obtienen unos resultados lo más independientes posibles.

Otro parámetro importante es el tamaño de los paquetes (L), puesto que de él dependen las tasas de envío. Al igual que en los anteriores, se considerará que $200 < L < 1500$ bytes (cabeceras incluidas) y además, también debe ser múltiplo de 92, para evitar el relleno de ceros (padding) que se auto genera en los enlaces ALL5-ATM, como ya se ha mencionado en anteriormente. Los valores más eficaces que se han encontrado han sido cuando los paquetes cumplen $L > 1000$ bytes [Rib+03].

En *PathChirp*, el estudio estadístico del OWD se realiza de manera diferente a como se ha realizado anteriormente. En este caso, una vez obtenidas los OWDs en cada flujo, se realiza un análisis en cada flujo en busca de lo que se llaman *excursiones* en el retardo de los paquetes. Para detectar estas *excursiones*, tan sólo es necesario medir la tendencia de los OWDs, es decir, si se experimenta una tendencia creciente o no. Si se experimenta una tendencia creciente, se evaluará si es debido a un retardo de la red propiciado por nuestros paquetes o de lo contrario, por qué la red está saturada.

[3] *chirp*: grupo de paquetes separados en el tiempo de forma exponencial que permite medir varias tasas simultáneamente.

Para la detección de una *excursión*, hay que definir un punto de inicio de la excursión. Ese punto será el que en el retardo comience a elevarse, es decir, en cada flujo se va evaluando la diferencia de los OWDs consecutivos y si dicha diferencia es mayor con respecto a la anterior o se comienza a evaluar la excursión. Esto se puede mostrar mejor gráficamente en la Figura 20, en donde vemos como a partir del punto s (punto de inicio), es cuando el OWD del tercer paquete es mayor que el del segundo, lo que indica que ha existido una tendencia creciente y es el comienzo de lo que puede ser una excursión.

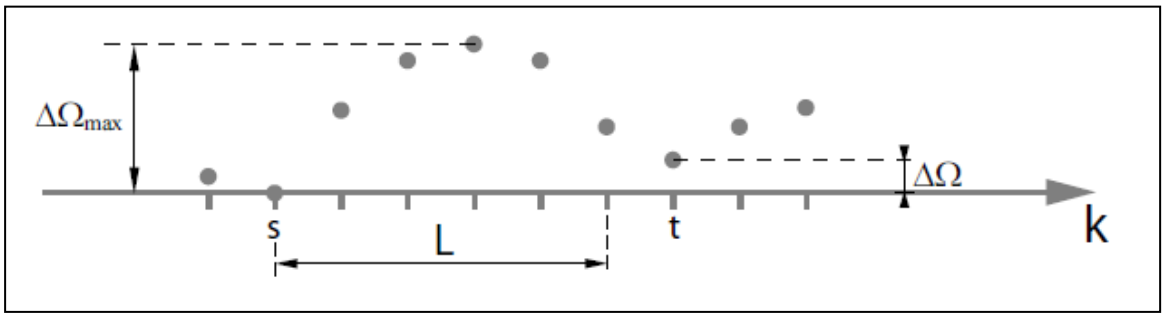

Figura 20. Detección de excursiones [CV06]

Después hay que encontrar, si lo hubiera, el punto de finalización t, qué indica que la excursión ha parado. Este punto t, se puede obtener mediante la expresión 3.20 y como se muestra en la Figura 20, es el punto en el que finaliza la primera excursión, puesto que es el punto donde el paquete siguiente vuelve a detectarse un incremento del OWD [CV06] Puede que este punto no llegue a existir y entonces querrá decir que la red está congestionada y se mostrará en el siguiente aparatado cuál será el ABw.

$$\Delta\Omega = \Omega_t^m - \Omega_s^m < \frac{\Delta\Omega_{\max}}{F} = \frac{max_{s\le k\le t}\{\Omega_k^m - \Omega_s^m\}}{F} \qquad (3.20)$$

Para la detección de estás excursiones, es importante definir dos parámetros más, el factor de decrecimiento F y la longitud de una excursión válida L. En 3.20 se puede ver que F indica en qué punto el OWD ha decrecido dicho factor. L es un umbral que determina la longitud máxima que debería tener una excursión para considerarla válida. En estudios realizados [Rib+03] se han fijado estos valores mostrados en la Tabla 4.

Parámetro	Valor por defecto	Valor con tráfico de internet
F	1.5	6
L	5	3

Tabla 4. Valores de F y L [Rib+03]

Proceso de medida y análisis de los resultados

El procedimiento para obtener el ABw en este algoritmo es el siguiente:

Una vez obtenidos los OWDs de cada paquete se procede a analizar si existe una excursión en cada uno de los flujos. Esto se realiza comparando los OWD consecutivos y en el momento en el que $\Omega_{k+1}^m - \Omega_k^m > 0$ se analizará si es una excursión válida. En ese momento queda determinado el punto de inicio de la excursión y por medio de la

expresión 3.20 se buscará el final de la excursión. En este punto, se pasará a realizar un proceso para estimar el ABw por paquete, en el que podrán suceder tres casos:

1) Si el paquete k pertenece a la excursión, quiere decir que se ha encontrado el punto final y la excursión debida a una pequeña congestión pero no elevada, con lo que aunque muestre una tendencia creciente, se estimará la propia tasa del paquete: $E_k^m = R_k^m$. Esto quiere decir que la posición de este paquete marca la tasa de envío.

2) Por otro lado, si el paquete pertenece a una excursión en la que no se ha encontrado el punto final, la estimación del ABw será la del paquete que señala el punto de inicio: $E_k^m = R_s^m$.

3) El tercer caso sería un punto intermedio entre los dos anteriores. Sucede cuando se detectan varias excursiones. Puede que las primeras sean pequeñas como las del primer caso y la última sea la que no tiene fin, como la del segundo caso. También puede suceder que la última sea infinita. En la primera de ellas, la estimación del ABw será la del punto de inicio de la última excursión: $E_k^m = R_{sl}^m$, y en la segunda, la tasa estimada será la correspondiente al retardo K-1, en este caso la más baja, $E_k^m = R_{k-1}^m$ puesto que no se ha encontrado el punto de finalización.

Una vez realizado este paso de estimación del ABw por paquete, se pasa a realizar la estimación por cada flujo, en donde se realiza una media ponderada de las estimaciones por paquete E_k^m calculadas en el paso anterior. Para ello se utilizará la expresión 3.21:

$$D^m = \frac{\sum_{k=0}^{K-2} E_k^m T_k^m}{\sum_{k=0}^{K-2} T_k^m} \tag{3.21}$$

donde T_k^mes el tiempo que existe entre el paquete k y el k+1 dentro del flujo m. De esta manera se obtienen la tasa media en cada flujo.

Finalmente, se realiza el promedio en base al número de flujos totales M, como se expresa en 3.22:

$$ABw = \frac{\sum_{m=0}^{M-1} D^m}{M} \tag{3.22}$$

En el apéndice 4 se muestra, en pseudo-código, un ejemplo de cómo puede ser el algoritmo.

3.2.3.2 Consideraciones en PathChirp [CV06]

Aunque el uso de los *chirps* es ventajoso puesto que en cada uno de ellos se puede mostrar todo el intervalo de medida, introducen algunas limitaciones en la resolución. Además, es complejo llegar a algunos valores óptimos como F y L para cualquier tipo de tráfico.

Resolución variable

Debido a la separación exponencial de los paquetes dentro del flujo, las diferentes tasas probadas tienden a concentrarse en torno a la parte baja del rango de medida, con lo que la resolución no es la misma en todo el rango y puede estimar a la baja en valores cercanos al límite superior del intervalo.

Libertad de elección de tasas intermedias limitada

Otra limitación derivada de la estructura de los *chirps* es que no deja la libertad de elegir tasas intermedias del intervalo. A modo de conclusión, PathChirp tiene mucha menos libertad en la resolución que TOPP y SLoPS. La resolución de estos dos métodos sólo depende del número de iteraciones.

Elección de los umbrales

F y L no están claramente determinados, de hecho, dependiendo de la red y del tipo de tráfico cruzado que exista en dicha red, se proponen unos valores u otros [Rib+03]. En la Figura 21 se muestra el error absoluto medido en una red con tráfico CBR, particularizando para diferentes valores de L y F. Se puede ver que para unos valores se produce una sobrestimación y para otros un subestimación.

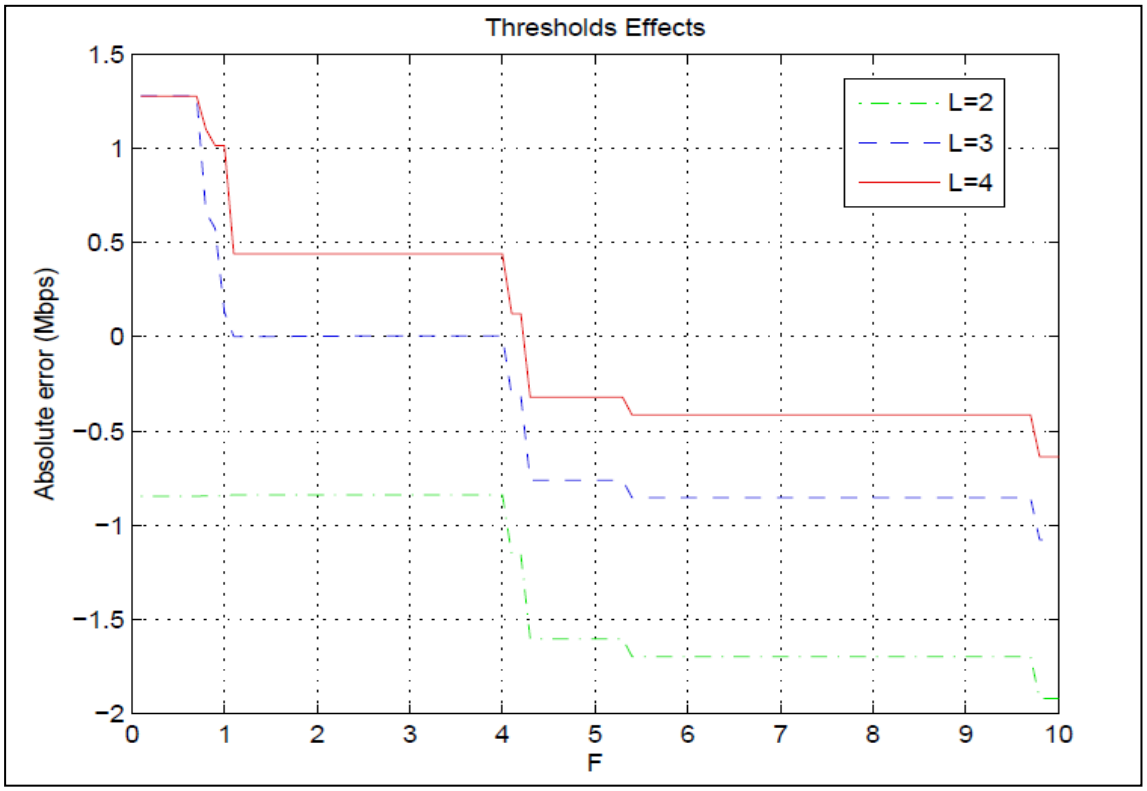

Figura 21. Error en PathChirp para diferentes valores de L y F [CV06]

3.3 Conclusiones

Los tres algoritmos escogidos son ejemplos de diferentes formas de poder obtener el ABw de una red extremo a extremo. Cada uno de ellos presenta características y propiedades diferentes. Para concluir, a modo de resumen, en la Tabla 5, se expondrán algunas de sus características más importantes, junto con algunos valores de sus parámetros y una comparación entre ellos.

TOPP	
Descripción	El algoritmo envía pares de paquetes equiespaciados en el tiempo, en cada iteración cambia el espacio tiempo entre paquetes para modificar la tasa.
Técnica	Packet Pairs (PP)
Alg. de búsqueda	Lineal
Medición	Relación entre entrada y salida (R_{in}/R_{out})
Intrusión	Muy Poca, aunque requiere mucho tiempo de prueba.
Sensibilidad al tráfico cruzado	Si
Precisión	Muy buena, aunque depende del número de iteraciones
SLoPS	
Descripción	SLoPS envía paquetes equiespaciados en el tiempo; en cada iteración va cambiando el tamaño de los paquetes, de forma que se modifica la tasa de envío.
Técnica	Auto-Congestión
Alg. de búsqueda	Binaria
Medición	OWD
Intrusión	Mucha, requiere muchos paquetes para obtener una medida precisa.
Sensibilidad al tráfico cruzado	No
Precisión	Muy buena, aunque depende los valores *w* y *X* que se establezcan.
PathChirp	
Descripción	El algoritmo envía *chirps*, que son paquetes espaciados entre sí que mantienen una relación exponencial de forma que se pueden medir varias tasas con sólo una iteración.
Técnica	Auto-congestión
Alg. de búsqueda	-
Medición	OWD
Intrusión	Poca.
Sensibilidad al tráfico cruzado	No
Precisión	Buena, aunque puede subestimar un poco el ABw y es menor que la de los dos algoritmos anteriores.

Tabla 5. Resumen de los tres algoritmos

Según algunas pruebas realizadas [Cas+06], algunos valores básicos óptimos para los tres algoritmos se muestran en Tabla 6.

Parámetro	TOPP	SLoPS	PathChirp
N	4	25	12
I	25	*	1
L (Bytes)	500	[200-1500]	1000
R (Mbps)	[1.0-9.0]	[1.0-7.5]	[1.1-7.9]

Tabla 6. Parámetros básicos de configuración [Cas+06]

Hay que destacar, que N es el número de muestras tomadas en la medida, por lo que tanto en SLoPS y PathChirp = K-1 y en TOPP es K/2.

El número de iteraciones en SLoPS es indeterminado por que al ser una búsqueda binaria puede variar, pero es inferior al que se necesita en TOPP.

Capítulo 4

Available Bandwidth Measurement Tool

En este capítulo se explicarán aspectos sobre la implementación y el desarrollo de los algoritmos explicados anteriormente, así como la visualización de los mismos dentro de la aplicación desarrollada. También veremos que se ha añadido otro algoritmo (que no está dentro de estas técnicas) a modo de comparación, puesto que consiste en enviar paquetes UDP y medir el tiempo que tardan en llegar. Como se verá el capítulo de resultados, se podrá utilizar para realizar una validación. Se le ha llamado UDPFile y se explicará su funcionamiento en el apartado 4.2.6.

En primer lugar se va describir brevemente la estructura del código mostrando diferentes diagramas y un recuento de las clases Java utilizadas.

Después se mostrará el Applet y una descripción de los componentes de dicho Applet de forma que se pueda tener una visión más clara del funcionamiento de la aplicación.

Además se visualizará cómo se muestran los resultados en la aplicación y una pequeña explicación de cómo interpretarlos.

Para terminar, se explicarán algunas modificaciones de los algoritmos realizadas a la hora de implementar la aplicación.

En el apéndice 1 se detalla un manual de funcionamiento más completo.

4.1 Estructura del código fuente

En este apartado se explicará cómo está estructurada la aplicación en cuanto a clases, así como un diagrama de clases y de secuencias.

4.1.1 Diagramas

El código fuente de la aplicación está constituido por 33 clases. Las clases principales están dividas entre el cliente y el servidor. Los diagramas de clases del cliente y del servidor de se muestran en la Figura 22 y Figura 23.

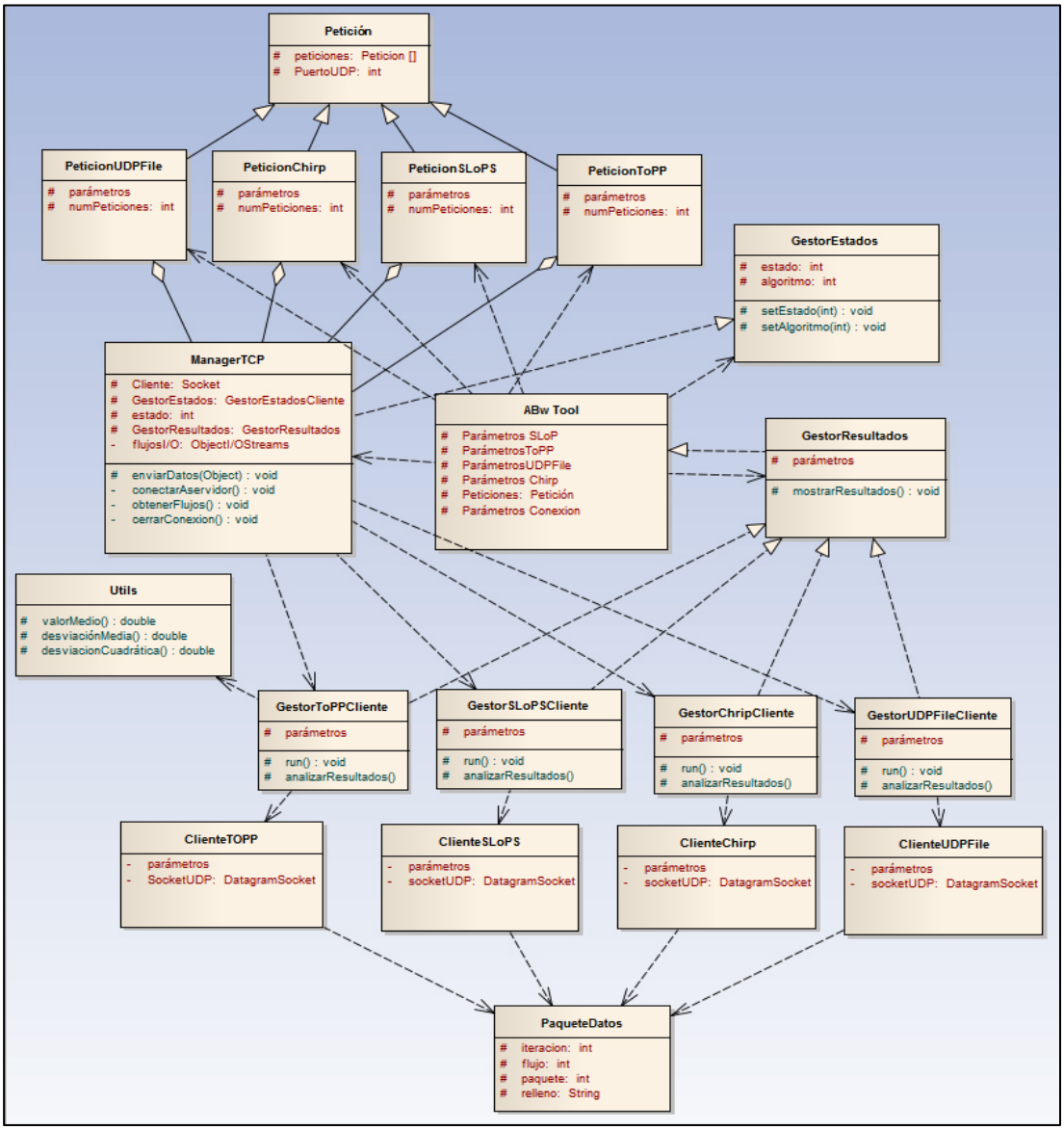

Figura 22. Diagrama de clases de la aplicación cliente

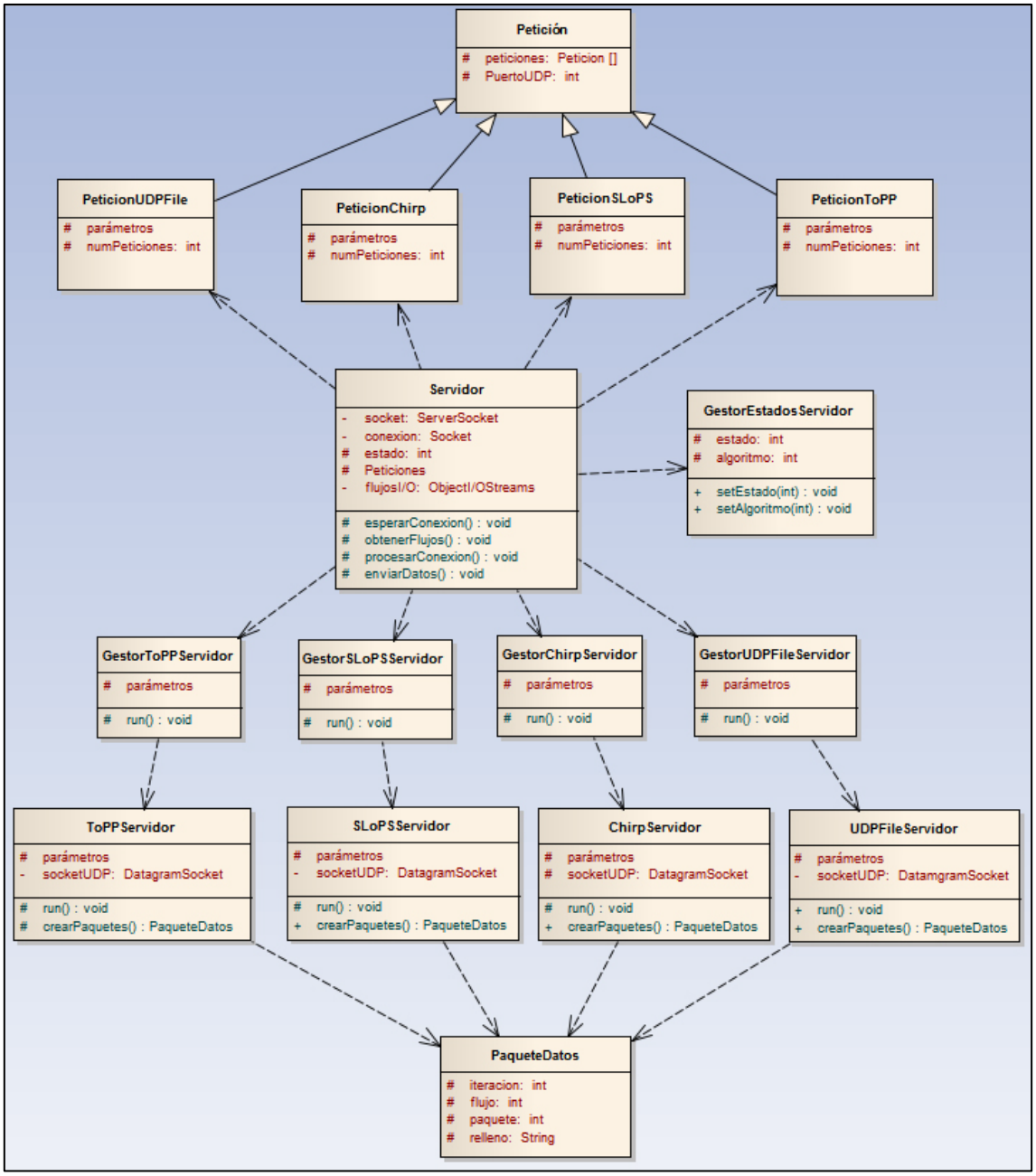

Figura 23. Diagrama de clases de la aplicación servidor

Para entender un poco el funcionamiento de la aplicación y el uso de las clases, en la Figura 24 se muestra el diagrama de secuencias, en el cual se puede observar el funcionamiento genérico de la aplicación para cada algoritmo.

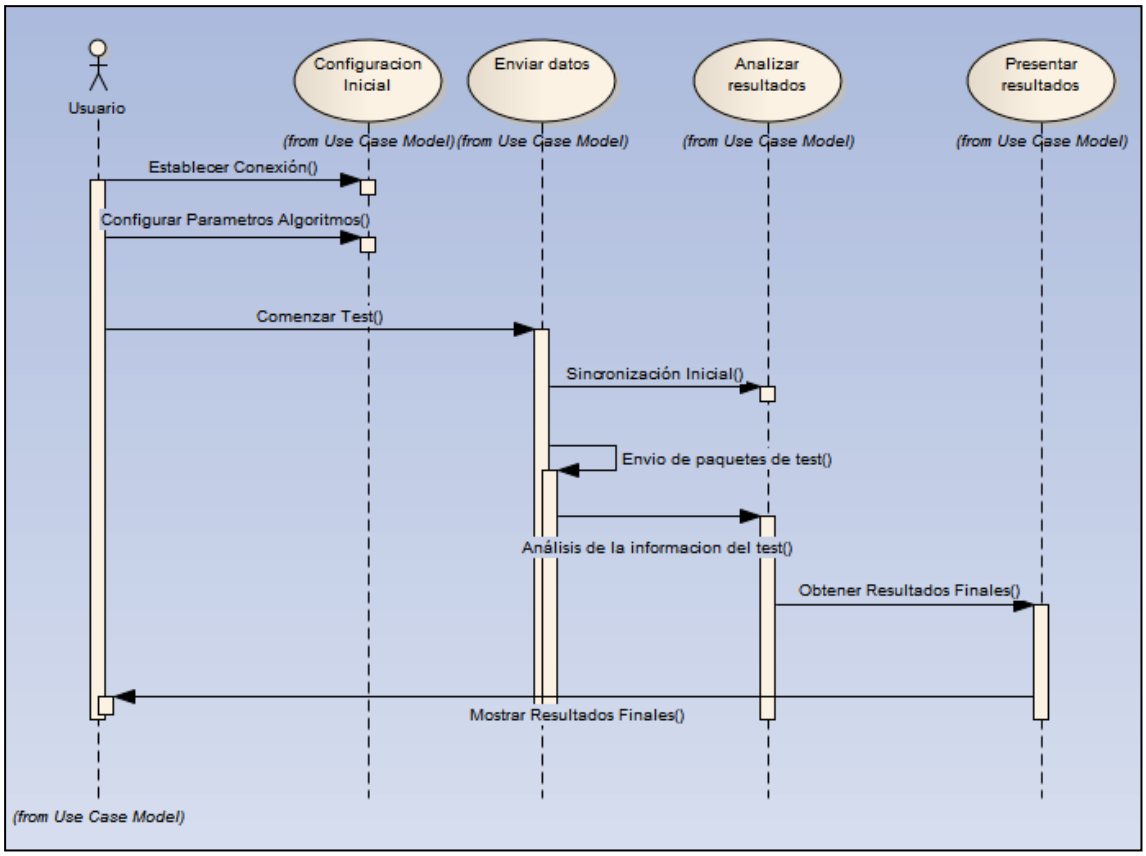

Figura 24. Diagrama de secuencias

Lo primero que se debe hacer es establecer la conexión con el servidor. Para ello se utiliza el panel conexión con el que se establecen los parámetros necesarios para llevar a cabo dicha conexión.

Una vez establecida la conexión, se procede a configurar el algoritmo con el cual se quiere medir el ABw. Es importante que todos los parámetros estén correctamente dispuestos para el buen funcionamiento del algoritmo. Después de haber establecido todos los parámetros se procede a ejecutar el algoritmo.

El algoritmo realizará una sincronización inicial entre cliente y servidor para poder iniciar el test. A continuación, se procederá a realizar el test, es decir, se empezarán a enviar los paquetes test para que fluyan por la red y lleguen al cliente.

Después de que el cliente reciba los paquetes, se procederá a su análisis para poder obtener los resultados.

Para finalizar, el cliente mostrará los resultados al usuario. Los resultados se mostrarán de forma que en el usuario se puedan comparar unos algoritmos con otros.

4.1.2 Recuento de líneas programadas

A modo orientativo, en la Tabla 7 se muestra un recuento del total de líneas que se han implementado para la realización de este proyecto.

Aplicación	Paquete (total paquetes)	Directorio (total dir.)	Archivo (total archivos)	Líneas
ABwTool	ABwTool.Cliente	Root	GestorEstadosCliente.java	≈ 235
			GestorResultados.java	≈ 775
			ManagerTCPCliente.java	≈ 644
		Chirp	ClienteChirp.java	≈ 330
			GestorChirpCliente.java	≈ 331
		SLoPS	ClienteSLoPS.java	≈ 326
			GestorSLoPSCliente.java	≈ 456
		TOPP	ClienteTOPP.java	≈ 316
			GestorTOPPCliente.java	≈ 136
		UDPFile	ClienteUDPFile.java	≈ 180
			GestorUDPFileCliente.java	≈ 57
	ABwTool.Servidor	Root	GestorEstadosServidor.java	≈ 89
			Servidor.java	≈ 360
		Chirp	ServidorChirp.java	≈ 130
			GestorChirpServidor.java	≈ 166
		SLoPS	ServidorSLoPS.java	≈ 160
			GestorSLoPSServidor.java	≈ 148
		TOPP	ServidorTOPP.java	≈ 159
			GestorTOPPServidor.java	≈ 144
		UDPFile	ServidorUDPFile.java	≈ 117
			GestorUDPFileServidor.java	≈ 99
	ABwTool.Comunes	Root	PaqueteDatos.java	≈ 116
			Peticion.java	≈ 65
		Chirp	ConfiguracionChirp.java	≈ 291
			PeticionChirp.java	≈ 122
		SLoPS	ConfiguracionSLoPS.java	≈ 305
			PeticionSLoPS.java	≈ 124
		TOPP	ConfiguraciónToPP.java	≈ 297
			PeticionToPP.java	≈ 137
		UDPFile	PeticionUDPFile.java	≈ 200
		Tools	Utils.java	≈ 43
	ABwTool.GUI	Root	ABwTool.java	≈ 3370
			DoubleSpinner.java	≈ 54
			IntegerSpinner.java	≈ 42
Total	**4**	**17**	**34**	**10524**

Tabla 7. Recuento de líneas de código

Cabe destacar que aproximadamente el 30% de las líneas corresponden a los archivos que conforman la interfaz gráfica y que es generado automáticamente por el *IDE Eclipse*.

4.2 GUI del Applet

La herramienta consta de seis paneles, uno de conexión, los cuatro paneles correspondiente a los algoritmos y un panel donde se muestran los resultados. Además se ha añadido un botón para ejecutar los cuatro algoritmos secuencialmente.

En la Figura 25 se puede ver el Applet completo con todos sus paneles.

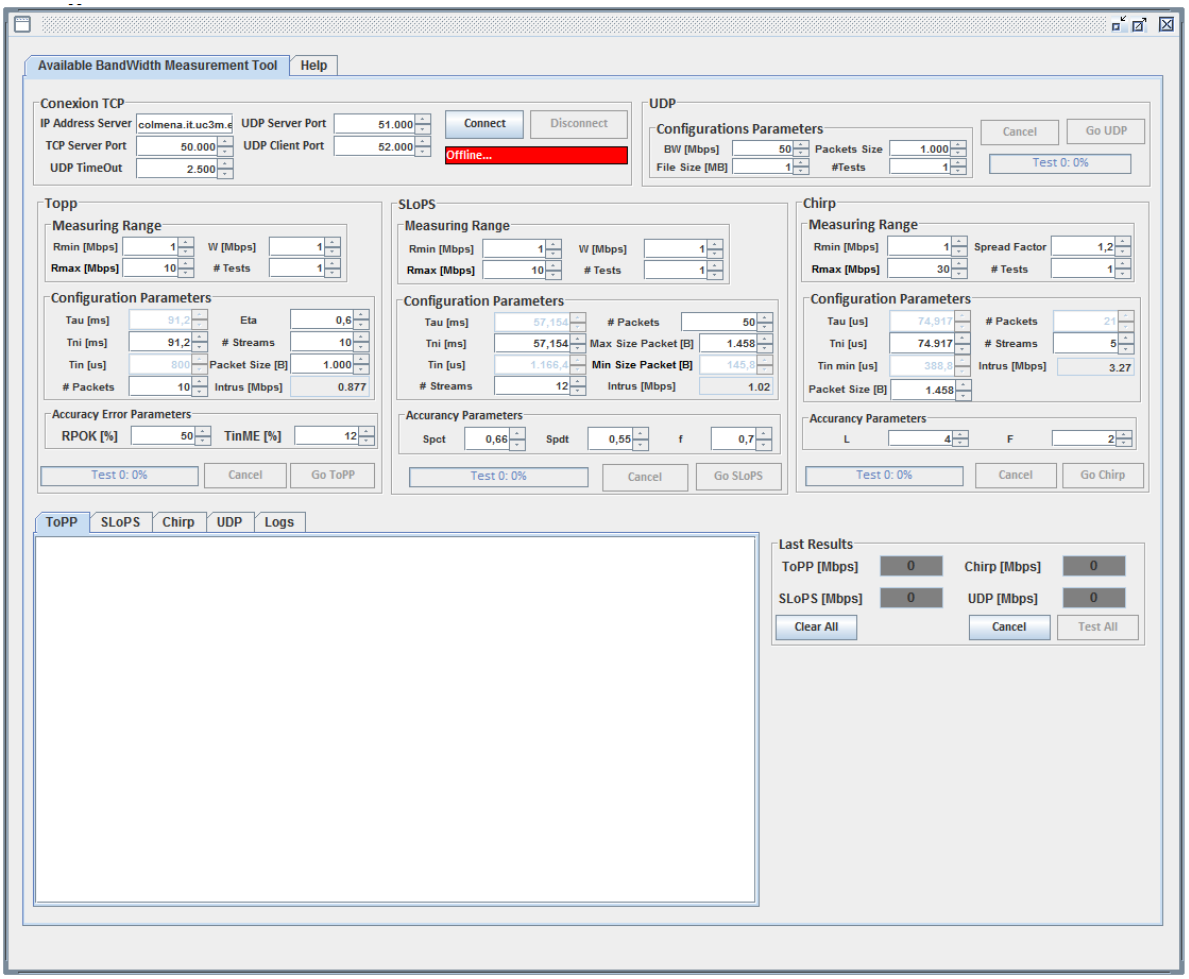

Figura 25. Available Bandwidth Measurement Tool

Como se puede ver, cada uno de los paneles de los tres algoritmos importantes está compuesto de tres sub-paneles:

- Sub-panel de rango de medidas: en el que se establece todo lo relacionado con las condiciones de la medida en general.
- Sub-panel de parámetros de configuración.
- Sub-panel de precisión de la medida.

Para el algoritmo *UDPFile* no existen diferentes sub-paneles ya que su funcionamiento es bastante más simple que los otros algoritmos.

A continuación se describirán brevemente algunos parámetros comunes y posteriormente se explicarán cada uno de los paneles de cada algoritmo.

4.2.1 Parámetros comunes

Como se ha podido ver en el capítulo 3, existen parámetros comunes a todos los algoritmos. Dichos parámetros se presentan en cada panel, algunos son meramente informativos y otros tan sólo son valores necesarios donde su valor no es tan determinante.

En la Tabla 8 se explican brevemente cada uno de dichos parámetros:

Parámetro	Definición
Tau	Tiempo que dura cada flujo.
T_{in}	Tiempo entre paquetes.
T_{ni}	Es un parámetro de seguridad para que no se junten demasiado los flujos entres sí, como ya ese ha explicado en capítulos anteriores.
Intrus	Indica el nivel de intrusión que mantiene el algoritmo, es decir, los bits por segundo reales que se transmiten en cada flujo. El número de paquetes y las tasas seleccionadas, varían bastante su valor.
Packet Size	Tamaño del paquete. Es bastante importante como ya se descubrirá más adelante. El tamaño máximo que se permite poner es de 1458 bytes, para dejar un pequeño margen con el valor máximo de MTU.
Número de Test	Es un parámetro con en el que se puede indicar cuantos test se quieren hacer. Los resultados se promediarán en base a ese valor.

Tabla 8. Algunos parámetros comunes a todos los algoritmos

4.2.2 Panel Conexión

Con este panel se configura y establece la conexión TCP y UDP. En este apartado se explicará el esquema principal de la arquitectura que se ha diseñado para la implementación de la aplicación.

El Applet se conecta mediante TCP al servidor. Mediante este protocolo se establece el control de la conexión y también se utiliza para la señalización, ya que es un protocolo orientado a conexión y tiene control de pérdidas de paquetes. De esta forma es más

seguro establecer la conexión. Además, también se utiliza para enviar ciertos datos utilizados para realizar algunos análisis de resultados como se verá más adelante.

La funcionalidad de TCP tanto en el cliente como en el servidor se muestra en la Tabla 9.

Función	Cliente	Servidor
Conexión	Siempre la establece el cliente.	Espera a recibir conexiones entrantes. Solo puede estar conectado a un solo cliente.
Solicitud de Peticiones	Envía la petición para realizar cada test.	El servidor atiende la petición y se prepara para comenzar el test.
Comienzo del test	El cliente avisa de que está listo para recibir los paquetes.	Una vez avisado por el cliente, comienza a enviar los paquetes mediante el protocolo UDP.
Estado durante el test	Entre cada iteración, se envía un mensaje de si se han recibido correctamente los paquetes con el fin de proseguir o abortar el test.	El servidor espera los mensajes de estado en cada iteración.
Envío Tiempos	Solicitud del envío de un vector con las marcas de tiempo del origen. (Sólo SLoPS y Chirp)	Cuando se solicita, se envía el vector con las marcas de tiempo del origen.
Fin test	Finalización del test.	Finalización del test.
Desconexión	Se desconecta del servidor.	Vuelve a esperar otra conexión.

Tabla 9. Funcionalidad del protocolo TCP

El protocolo UDP, como ya se ha mencionado en capítulos anteriores se utiliza para enviar los paquetes de prueba para llevar a cabo la medida.

En la Figura 26 se muestra dicho panel con sus parámetros.

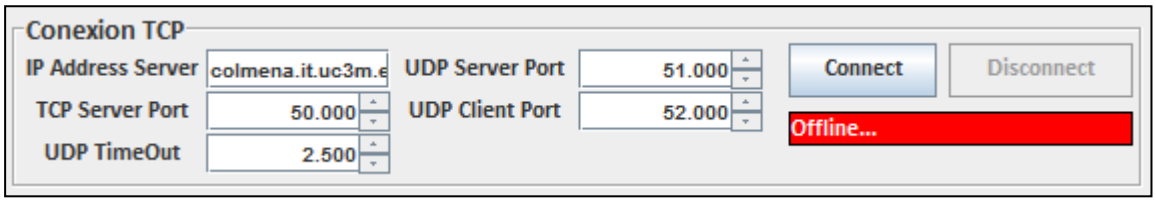

Figura 26. Panel Conexión

Los parámetros de este panel son los siguientes:

Parámetro	Definición
IP Address Server	Dirección IP del servidor
TCP Server Port	Puerto TCP en el que escucha el servidor.
UDP Server Port	Puerto UDP en el que escucha el servidor.
UDP Client Port	Puerto UDP en el que escucha el cliente.
UDP Time Out	Tiempo máximo que espera el cliente para recibir los paquetes.

Tabla 10. Parámetros panel Conexión

Además de los parámetros, hay una barra de estado indicando la situación de la conexión en cada momento. Los valores de los estados son los siguientes:

Estado	Significado	Color
Offline	Desconectado	Rojo
Connecting	Intentando establecer la conexión.	Amarillo
Connected	Conectado	Verde

Tabla 11. Valores de estado de la conexión

Los valores por defecto de los puertos se han elegido de forma que se utilicen puertos que no son utilizados habitualmente por aplicaciones más usuales. En este caso se ha decido usar para TCP el puerto 50000 y para UDP el 51000. Esto está justificado de acuerdo con la tabla que propone IANA.

4.2.3 Panel TOPP

En la Figura 27 se puede ver el panel con el que se puede configurar y lanzar TOPP.

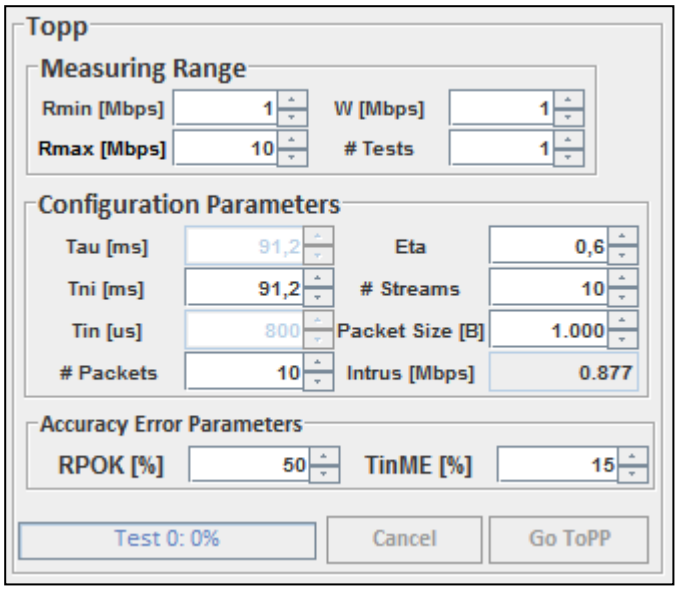

Figura 27. Panel TOPP

Además de los parámetros comunes ya mencionados en el primer apartado, en la Tabla 12 se definen brevemente los más importantes.

Parámetro	Definición
R_{min} [Mbps]	Es la tasa mínima a medir. Es importante establecer una tasa mínima que sea inferior al ancho de banda hipotético.
R_{max} [Mbps]	Tasa máxima, si se sabe que no existe más de un cierto ABw se debe fijar a un valor cercano para no tener muchas iteraciones.
W [Mbps]	Resolución que se quiere obtener. Es el paso de medida en cada iteración.
Eta	Valor que indica el valor mínimo de T_{pp} con respecto a T_{in}.
T_{in}ME [%]	Margen de tiempo, en porcentaje, entre el que deben de estar los paquetes separados para que se cumplan las condiciones de ABwE.
RPOK [%]	Porcentaje de paquetes que entran el margen de tiempo establecido.

Tabla 12. Parámetros más relevantes de TOPP

4.2.4 Panel SLoPS

En la Figura 28 se muestra el panel con el que se puede configurar SLoPS y lanzarlo.

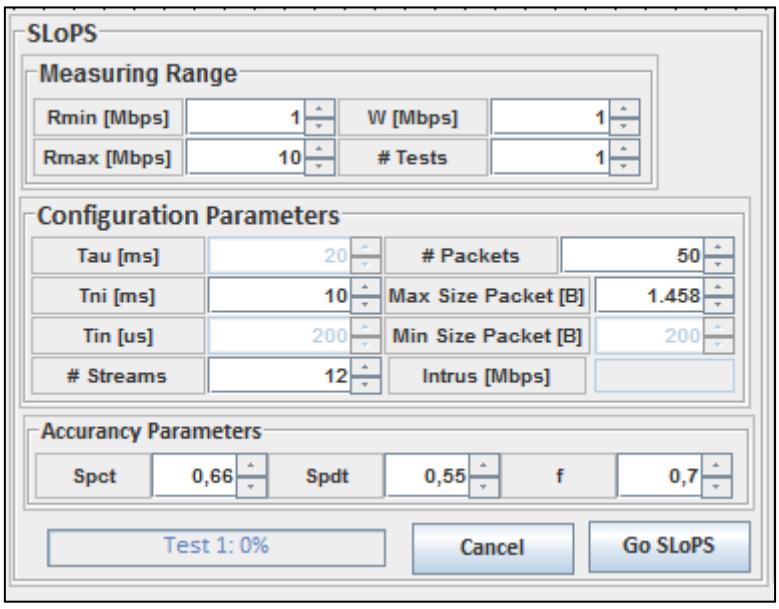

Figura 28. Panel SLoPS

En la Tabla 13 se explican brevemente los parámetros más importantes de SLoPS.

Parámetro	Definición
R_{min} [Mbps]	Tasa mínima a medir. Es importante establecer una tasa mínima que sea inferior al ancho de banda hipotético.
R_{max} [Mbps]	Tasa máxima, si se sabe que no existe más de un cierto ABw se debe fijar a un valor cercano para no tener muchas iteraciones.
W [Mbps]	Resolución que se quiere obtener.
S_{pct}	Parámetro con el que se determina si se ha producido una tendencia creciente o decreciente.
S_{pdt}	Parámetro con el que se determina si se ha producido una tendencia creciente o decreciente.
f	Umbral con el que se determina si se han producido más tendencias crecientes que decrecientes.

Tabla 13. Parámetros más relevantes de SLoPS

4.2.5 Panel Chirp

En la Figura 29 se muestra el panel con el que se puede configurar Chirp y lanzarlo.

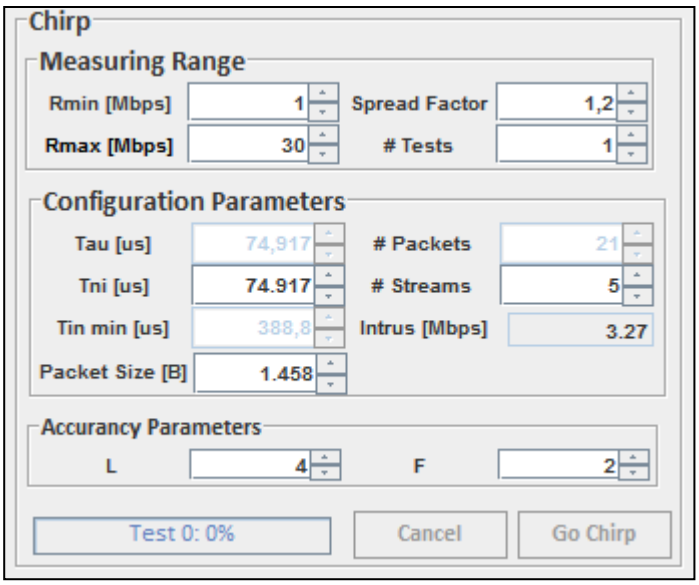

Figura 29. Panel Chirp

Los parámetros más importantes se definen brevemente en la Tabla 14. Parámetros más relevantes de Chirp:

Parámetro	Definición
R_{min} [Mbps]	Tasa mínima a medir. Es importante establecer una tasa mínima que sea inferior al ancho de banda hipotético.
R_{max} [Mbps]	Tasa máxima, si se sabe que no existe más de un cierto ABw se debe fijar a un valor cercano para no tener muchas iteraciones.
Spread Factor	Factor de expansión, es el valor del exponente para llevar a cabo la variación exponencial de los tiempos.
L	Tamaño de una excursión válida.
F	Factor de decrecimiento para detectar que una excursión ha terminado.

Tabla 14. Parámetros más relevantes de Chirp

4.2.6 Panel UDPFile

En la Figura 30 se muestra el panel del algoritmo UDP.

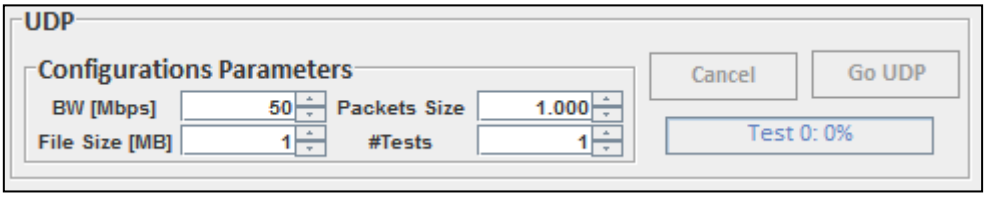

Figura 30. Panel UDPFile

Los parámetros más importantes se muestran en la Tabla 15 :

Parámetro	Definición
BW	Ancho de banda máximo que se quiere medir.
File Size [MB]	Tamaño en MB de la cantidad de datos que se quiere enviar.

Tabla 15. Parámetros más relevantes de UDPFile

4.2.6.1 Funcionamiento de UDPFile

Debido a que este algoritmo no es más que una pequeña herramienta de validación, su funcionamiento no se ha explicado anteriormente. Por ello se va a explicar en este apartado aprovechando que se ha mostrado su panel y sus parámetros.

UDPFile se basa en el envío a una tasa media determinada de paquetes entre el servidor y el cliente. El ABw se estima midiendo la relación entre la cantidad de bytes recibidos y el tiempo invertido en el envío, pero no se cuenta el tiempo de transmisión, ni retardos. Solo se tiene en cuenta lo que el cliente recibe. Se divide la cantidad de bytes recibidos entre segundos empeñados. Este funcionamiento tan simple hace que el algoritmo necesite enviar muchos paquetes con lo que eleva el nivel de intrusión y además, la red nota bastante dicha congestión.

Aunque no es una medida demasiado precisa, se comprobará en el capítulo de resultados que es bastante útil para realizar una rápida y pequeña validación. Además es bastante parecido al Iperf, solo que es más sencillo.

4.2.7 Panel Resultados

En este apartado, se visualizarán los resultados y una breve explicación de cómo se muestran en cada algoritmo para poder analizarlos correctamente.

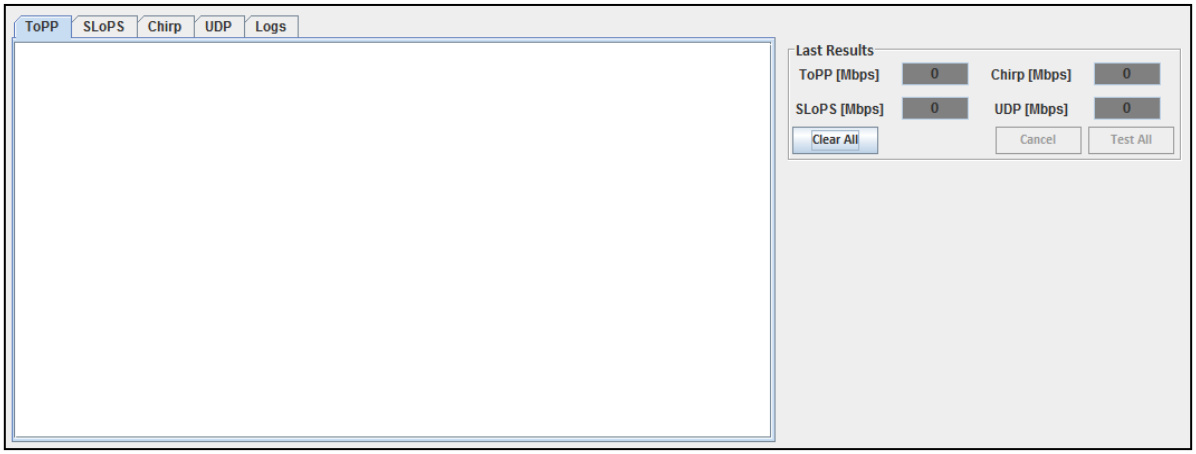

Figura 31. Panel donde se muestran los resultados

En cada pestaña se expondrán los resultados de cada uno de los algoritmos correspondientes. Además, en el panel de últimos resultados se mostrará la última prueba realizada y se marcará en amarillo la más alta de las cuatro.

Cada uno de los algoritmos tiene una forma diferente de mostrar los resultados. Por ello se va a explicar brevemente el resultado mostrado en cada pestaña.

4.2.7.1 Pestaña TOPP

En la Figura 32 se muestra como se presentan los resultados de TOPP.

En primer lugar, se muestra la configuración del algoritmo, es decir, los valores de los parámetros seleccionados para dicha medida.

A continuación se muestran los resultados de cada iteración. Estos resultados están constituidos por la tasa de envío, la tasa con la que se recibe, la relación entre tasa enviada y tasa recibida, número de paquetes que llegan en el margen de tiempo que deberían de llegar para considerar que no ha existido congestión y porcentaje de error de dichos paquetes.

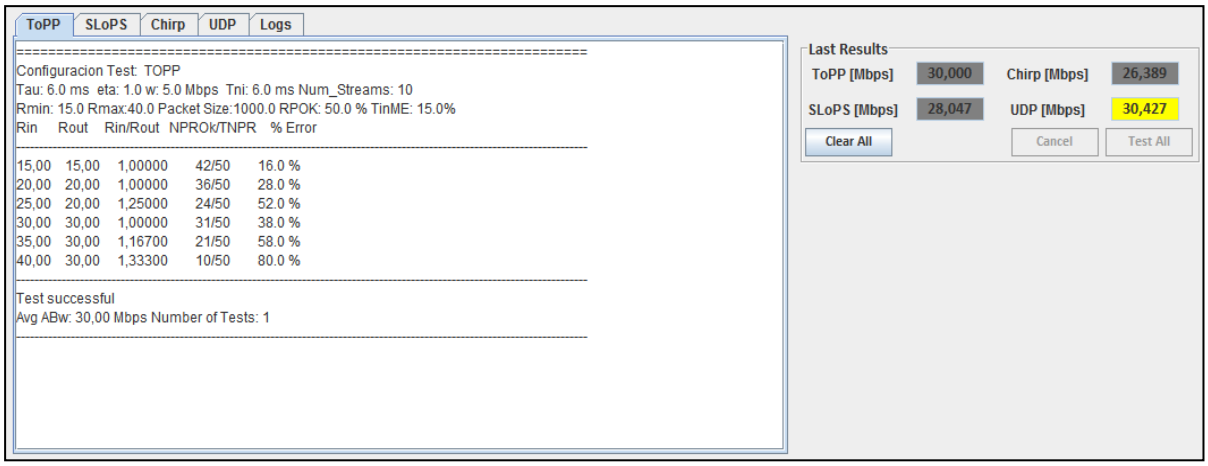

Figura 32. Resultados pestaña TOPP

Finalmente, se muestra el resultado final, acompañado del número de test que sea han realizado en esa prueba y se actualiza el valor del panel de los últimos resultados.

También se puede ver el resultado en el panel de últimos resultados. Como en ese momento no es el mayor de todos, no se muestra en amarillo.

4.2.7.2 Pestaña SLoPS

En la Figura 33 se muestra como aparecen los resultados de SLoPS.

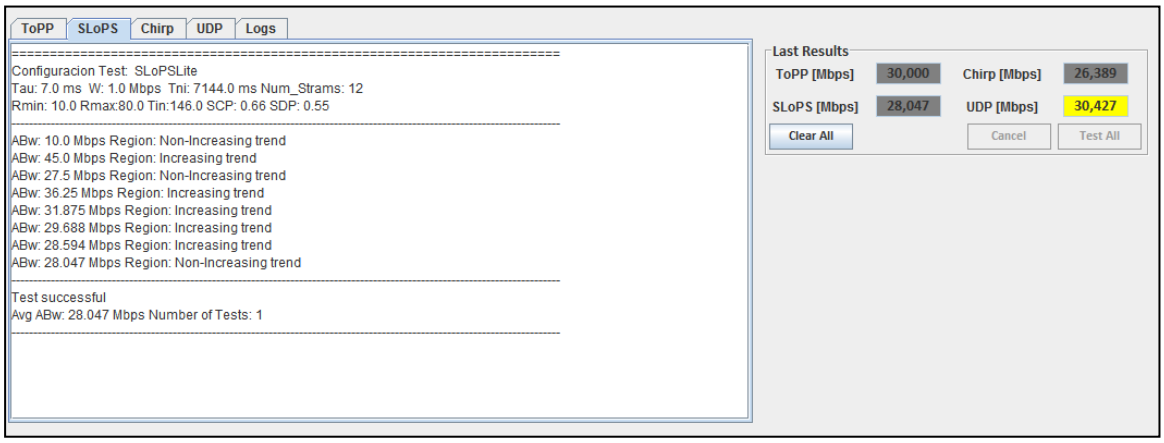

Figura 33. Resultados pestaña SLoPS

Al igual que en TOPP, lo primero que se muestra es la configuración de la prueba.

Después, aparecen los resultados de cada iteración. En este caso, el resultado está comprendido por la tasa de envío y la región a la que corresponde, es decir, si ha sufrido una tendencia creciente o decreciente en el camino.

Finalmente, se muestra el resultado obtenido acompañado del número de test que se han realizado en esa prueba.

Se puede ver también, que se actualiza el valor en el panel de últimos resultados, pero como no es mayor que los que están, no se ha resaltado en amarillo.

4.2.7.3 Pestaña Chirp

En la Figura 34 se muestra como aparecen los resultados de PathChirp.

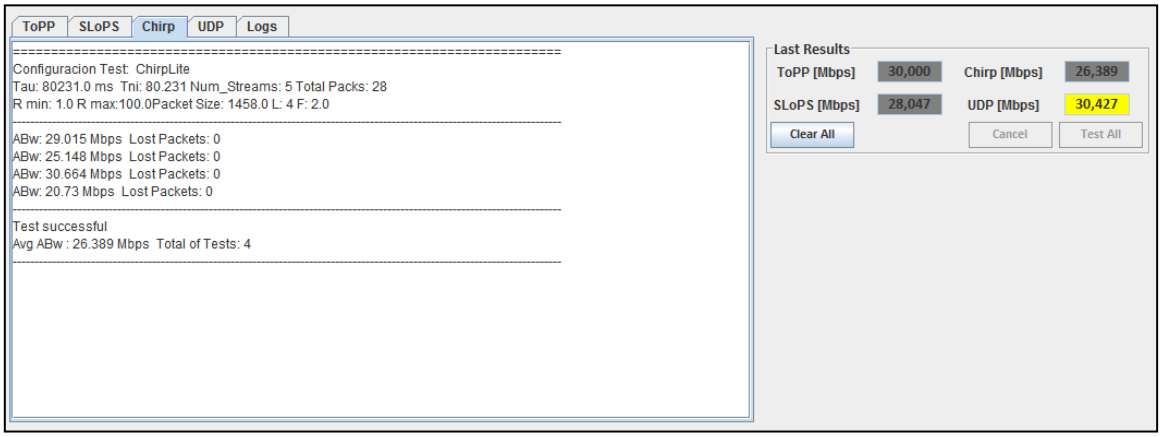

Figura 34. Pestaña resultados PathChirp

En primer lugar, igual que en los dos anteriores, se muestra la configuración inicial.

Después se muestran los resultados. En este caso, el algoritmo solo realiza una iteración, con lo que muestra el resultado de esa iteración junto con los paquetes perdidos. En la Figura 34 se ha completado con cuatro test en total.

Finalmente se muestra la media del resultado obtenido en los test realizados y el número de pruebas realizadas.

Nuevamente, se actualiza el panel de últimos resultados.

4.2.7.4 Pestaña UDP

En la Figura 35 se muestra como aparecen los resultados en la pestaña UDP.

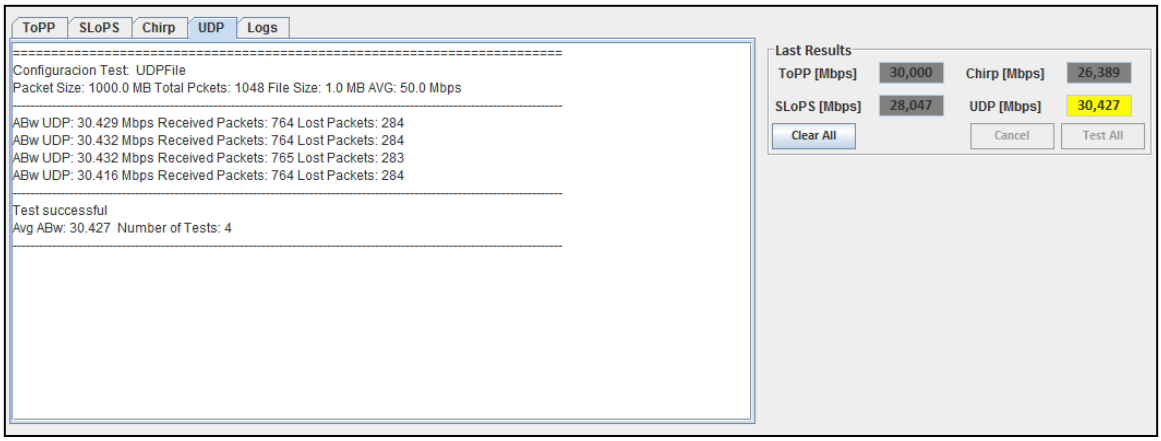

Figura 35. Pestaña resultados UDP

De la misma forma que los anteriores, la primera parte corresponde a la configuración del algoritmo.

A continuación, al igual que en *PathChirp*, como solo hay una iteración, se muestra la iteración, junto con el número de paquetes en recibidos y perdidos.

Finalmente se muestra el resultado total junto con el número de test realizados.

También se actualiza el panel de últimos resultados y como en este caso, es el valor más alto, se resalta en amarillo.

4.2.7.5 Pestaña Logs

En esta pestaña se resumen todas las pruebas de los algoritmos que se han ejecutado. En la Figura 36 se puede observar cómo se muestra dicho resumen.

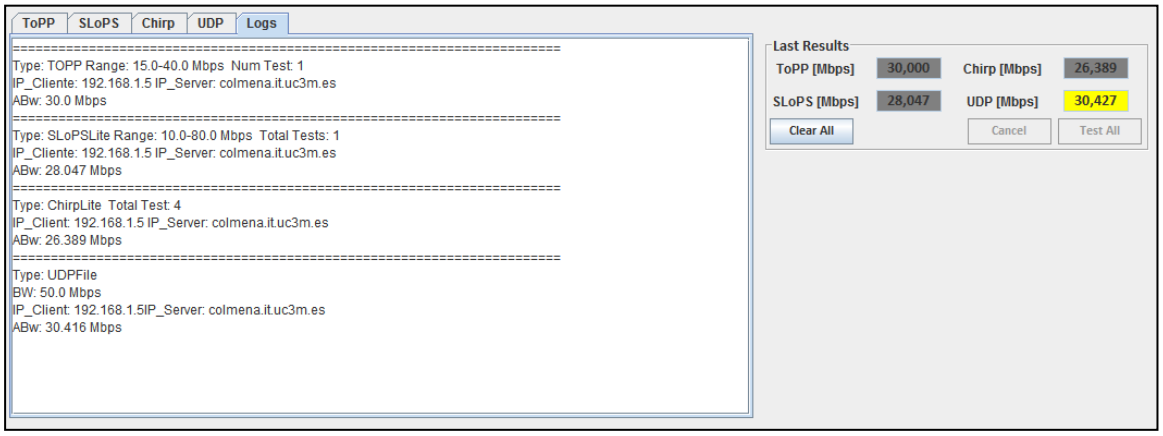

Figura 36. Pestaña resultados Logs

4.3 Connotaciones importantes

A la hora de implementar los algoritmos, se han realizado una serie de modificaciones en algunos algoritmos con el fin de simplificar su implementación. En este apartado se explicarán esas modificaciones realizadas en cada uno de los algoritmos y se justificarán dichas modificaciones. Dichas modificaciones se han realizado en los algoritmos TOPP y SLoPS.

4.3.1 Modificaciones realizadas en TOPP

En el capítulo 3 se explicaron algunas formas de analizar los resultados de TOPP. Para simplificar dichos análisis se ha modificado un poco la forma en que se calcula el ABw.

El proceso que se ha seguido es el siguiente:

En primer lugar, una vez que se tienen todas las marcas de tiempo de los paquetes recibidos, se comprueba cuales de los pares de paquetes llegan con la separación con la que deberían llegar. Es decir, si están en el margen establecido por el parámetro T_{in}ME. Además se calcula qué porcentaje de dichos paquetes llegan en ese margen. Con esto se decide si la medida es correcta o no. Es decir, si en esa iteración R_{in} > ABw o R_{in} =< ABw. De esta forma se puede deducir si hay congestión o no, y a que tasa.

Para poder justificar esta solución, se realizaron algunas pruebas que se mostrarán a continuación.

Estas pruebas se han realizado en Ethernet, para ver cómo llegan los paquetes separados y poder establecer un máximo en la medida.

En primer lugar, en la Figura 37 se muestra la dispersión de tiempos que se produce en varias tasas. Esto nos indica hasta cuanto ABw se puede llegar a medir de forma correcta con Java.

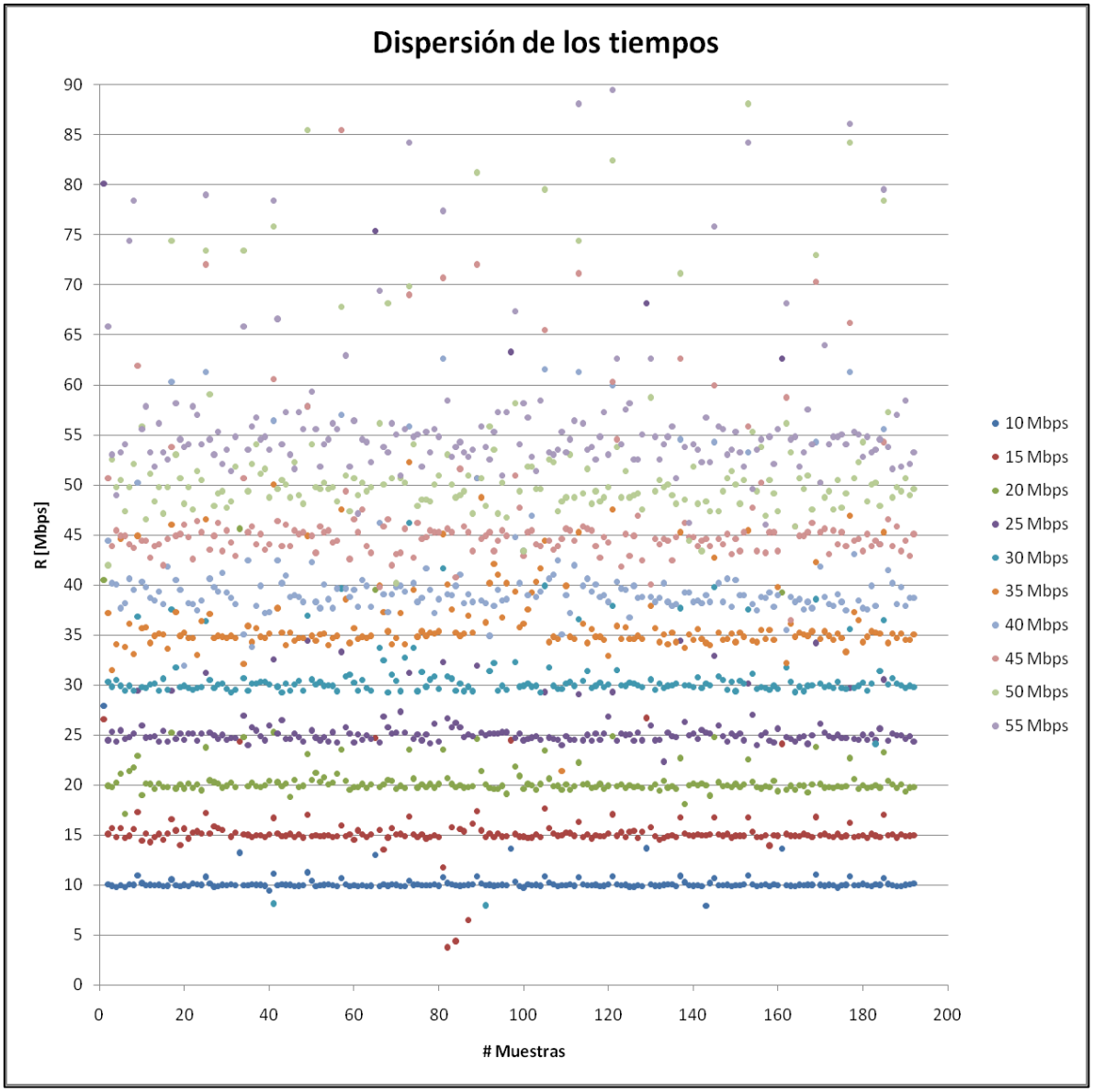

Figura 37. Dispersión de tiempos en TOPP

Se puede ver que hasta 35 ó 45 Mbps, las medidas pueden ser bastante precisas. Se puede observar que prácticamente los tiempos recibidos entre 10 y 35 son muy buenos puesto que se corresponden con su tasa enviada. A partir de esos valores, los tiempos comienzan a dispersarse y ya no hay tanta precisión. Por ello, se ha realizado otra prueba en la que se mide una tasa de 60 Mbps. Esto se muestra en la Figura 38, donde se puede observar que para una tasa de envío de 60 Mbps, la mayoría de los paquetes llegan con tiempos espaciados dentro de un margen del 15 % con respecto a la separación con la que se ha enviado. Esto indica, que el 15 % es un valor umbral válido para obtener la medida.

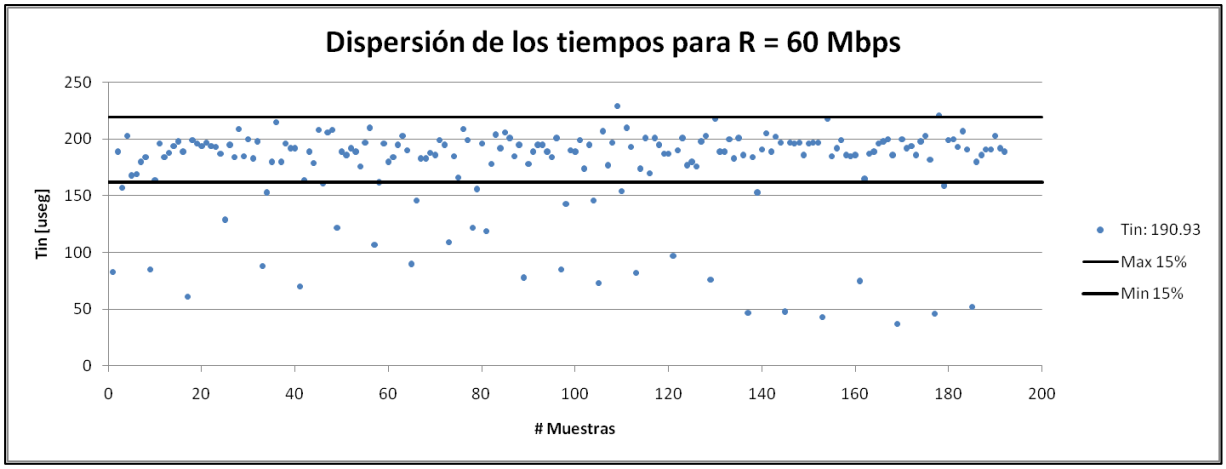

Figura 38. Dispersión de tiempos para R = 60 Mbps

Además, se ha obtenido un intervalo de confianza al 95 % de 5.68 useg., lo que indica que en la mayoría de las ocasiones, no se sobrepasa una diferencia de un 3 %.

El único problema es que no se podrán distinguir tasas demasiado cercanas entre sí. Por ello, se verá que en la parte de resultados, ese margen de 15 % es variable en la aplicación y se podrá ajustar a valores óptimos en función de la medida que se esté realizando.

4.3.2 Modificaciones realizadas en SLoPS

En SLoPS, se ha realizado una pequeña modificación cuando se detecta la región a la que pertenece cada iteración. Como se explicó en el capítulo 3, el algoritmo obtenía el ABw en función de la región en la se detectaban los paquetes enviados. Estas eran tres: Región creciente, decreciente y una región intermedia denominad región gris. La modificación se basa en la eliminación de dicha región gris. De esta forma se simplifica el algoritmo y además se evitan algunos problemas surgidos cuando no se detectaba bien esa región.

El motivo por que cual se ha decidido esta reducción de regiones está precedido por el problema del apartado anterior, donde la precisión no es la idónea. En numerosas ocasionas ocurría que al determinar una región, si en primera instancia se detectaba una región gris y posteriormente se detectaba una región no creciente, el algoritmo no estaba

preparado para ese caso y se obtenían resultados incoherentes. Al quitar la región gris, se cambia la resolución dentro de esa región, es decir, dentro de esa región, el parámetro *x* fijaba la resolución. Al quitar dicha región, ya no hay que establecer una resolución para dicha región. De esta manera, cuando se detectaban varias tasas dentro de la región gris, se buscaba la mayor de todas, pero siempre esperando encontrarse en algún momento con una región creciente para poder finalizar. Como esta condición no ocurría siempre, se decidió eliminar dicha región.

Esta modificación se ha completado de forma que el valor por defecto de la región sea una región no creciente. De esta forma, siempre se busca la máxima tasa posible.

Capítulo 5

Resultados

En este apartado se mostrarán los resultados obtenidos en diversas simulaciones realizadas para diferentes escenarios.

Para cada escenario se presentará una descripción del mismo, seguida de las pruebas realizadas en dichos escenarios.

En la medida que sea posible se realizará una validación con una aplicación llamada Iperf que también utiliza el protocolo UDP para la obtención de la medida, que es bastante precisa. En otros casos, se utilizarán otras herramientas para realizar la validación. De esta forma se podrá ver la precisión de las medidas.

En todas las pruebas se ha realizado un promedio sobre 20 medidas para obtener el resultado de la tendencia de la medida.

5.1 Escenario controlado de referencia

En la Figura 39 se puede ver un ejemplo de una red con un solo salto. El enlace del usuario está limitado a 10 Mbps.

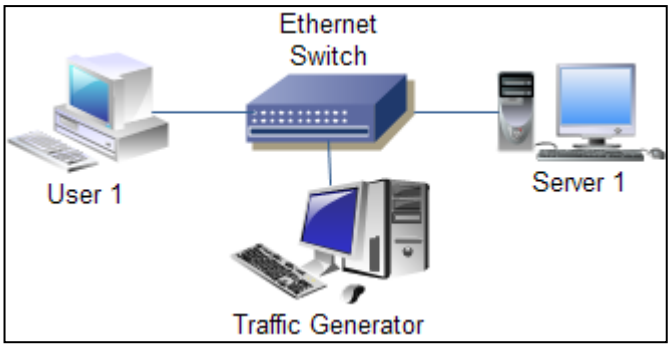

Figura 39. Escenario controlado

El hardware empleado para este escenario se reflejan en la Tabla 16:

Parámetro	Servidor	Cliente	Generador de Tráfico
CPU	Intel Core 2 Duo 2100	Intel Core 2 Quad 2600	Pentium IV 2800
RAM	4 GB	4 GB	3 GB
Sistema Operativo	Linux Ubuntu 10.4	Windows 7	Linux Ubuntu 10.4
JVM	Versión 6.21 32 bits	Versión 6.19 64 bits	Versión 6.21 32 bits
Navegador	-	Mozilla y Chrome	-

Tabla 16. Hardware utilizado para el escenario simple

A continuación se van a mostrar los resultados de una medida en un escenario simple sin tráfico cruzado para cada uno de los algoritmos y validándolo con Iperf. Seguidamente se realizará la misma operación añadiendo tráfico cruzado del tipo CBR. Dicho tráfico cruzado se va a generar con una herramienta llamada D-ITG (Distributed Internet Traffic Generator) [DITG]. Con esta herramienta se puede generar un tráfico de tasa constante CBR (Constant Bit Rate), de forma que en una enlace de 10 Mbps circule un tráfico de X Mbps y el ABw = C - X Mbps.

5.1.1 Resultados de TOPP

Con TOPP se han realizado diferentes tipos de mediciones. Se ha probado su capacidad para obtener el ABw en función de varios parámetros como puede ser el tamaño de los paquetes utilizados, el número de paquetes por flujo (K) y *Eta*. Las pruebas que se han realizado para el escenario de la Figura 32 han sido las siguientes:

Enlace sin congestión, capacidad 10 Mbps

En la Figura 40 se muestra las medidas realizadas en función del tamaño del paquete

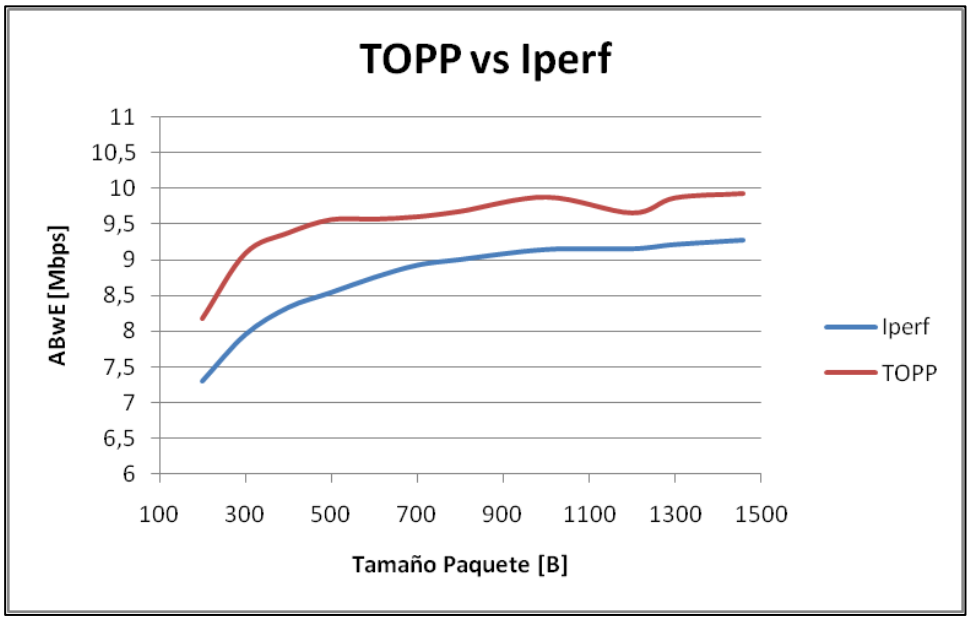

Figura 40. TOPP vs Iperf en función del tamaño del paquete

Se puede comprobar que TOPP da un resultado un poco más elevado que Iperf, lo que indica que sobrestima la medida. Esto puede deberse al margen que se ha dejado en la precisión aunque, como veremos después, en todas la medidas TOPP da un resultado un poco más elevado que Iperf.

Por otro lado, cabe destacar que el resultado del ABw depende del tamaño del paquete utilizado. Es decir, lo ideal es utilizar un tamaño de paquete entre 800 y 1500 bytes.

Además, el error que se comete con TOPP con respecto a Iperf en esa franja no supera el 8,2%.

Para ver el rendimiento en función del número de paquetes por flujo, se ha realizado el test mostrado en la Figura 41. En ella se puede comprobar que el parámetro K no es muy relevante.

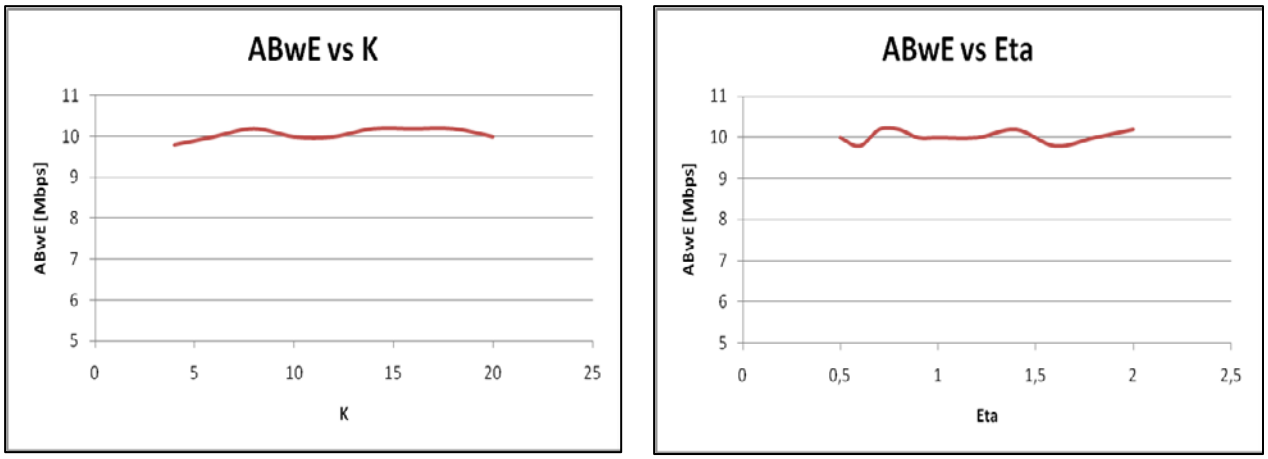

Figura 41. TOPP en función de K (izqda.) y *Eta (dcha.)*

De la misma forma, se ha evaluado el rendimiento con respecto a *Eta*.
Al igual que con K, en la Figura 41 se puede ver que *Eta* tampoco es un parámetro demasiado relevante. Pero sí es importante que no sea muy elevado puesto que de él depende el parámetro Tau y conviene que tenga un valor pequeño.

Enlace con tráfico cruzado a una tasa constante (CBR) a 4 Mbps y Lx = 800 B.

En la Figura 42 se muestran los resultados de las medidas obtenidas inyectando un tráfico CBR con una tasa de 4 Mbps y con un tamaño de paquete de 800 Bytes.

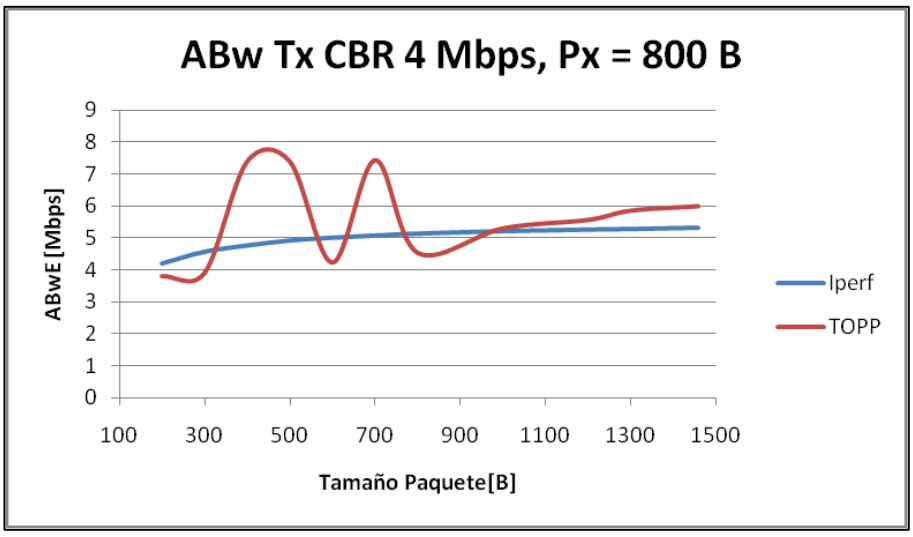

Figura 42. TOPP vs Iperf con T_x en función del L y Lx = 800B

Se puede observar que a partir de que el tamaño de los paquetes de prueba son menores que los paquetes del tráfico cruzado, el algoritmo empieza a perder efectividad e incluso, oscila bastante. Esta dependencia del algoritmo ya se comentó en el apartado 3.2.1.2.

Para poder comprobar esto con más detalle, se ha probado con diferentes tamaños de tráfico cruzado.

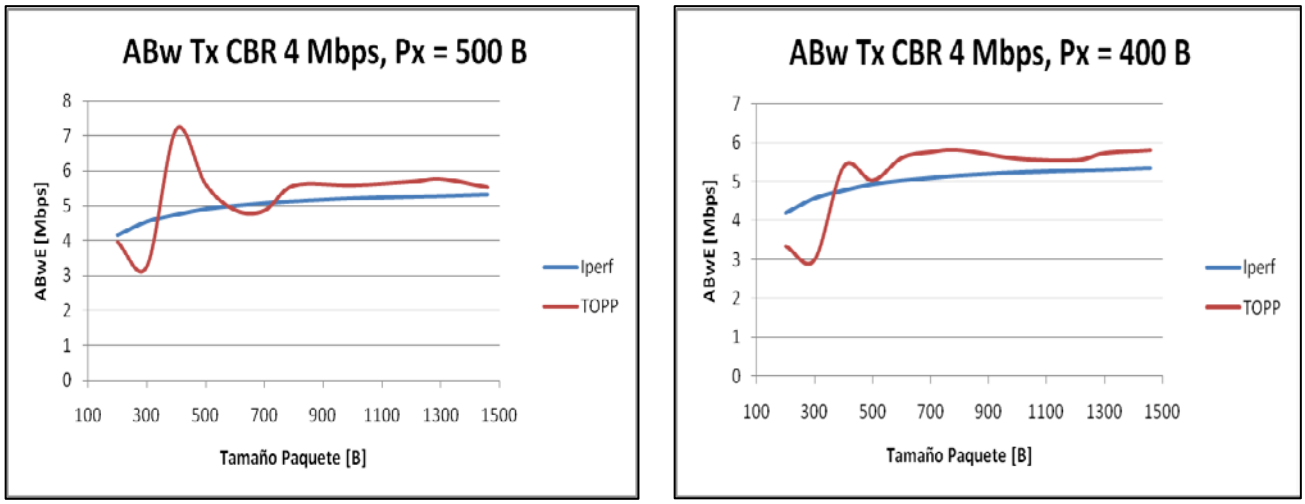

Figura 43. TOPP vs Iperf con T_x en función L, Lx = 500B. (izqda.) Lx = 400B. (dcha.)

En la Figura 43, se comprueba que en el momento en que el tamaño del paquete de prueba supera el tamaño del paquete de tráfico cruzado, el algoritmo funciona de forma estable.

5.1.2 Resultados de SLoPS

SLoPS parece tener un comportamiento bastante bueno. En la Figura 44 se muestran dichos resultados. Para realizar esta prueba, se ha utilizado el generador de tráfico [DITG]. Se puede comprobar que Iperf comete errores cuando el ABw está entre 2 y 4 Mbps.

Esto puede deberse a que el tráfico cruzado generado no es tan constante. Además, los dos algoritmos son bastante intrusivos y pueden afectar en bastante medida al tráfico que circula por la red, lo que puede hacer que el tráfico que se utiliza para medir, desplace al tráfico cruzado de forma que deje libre un poco más de ABw.

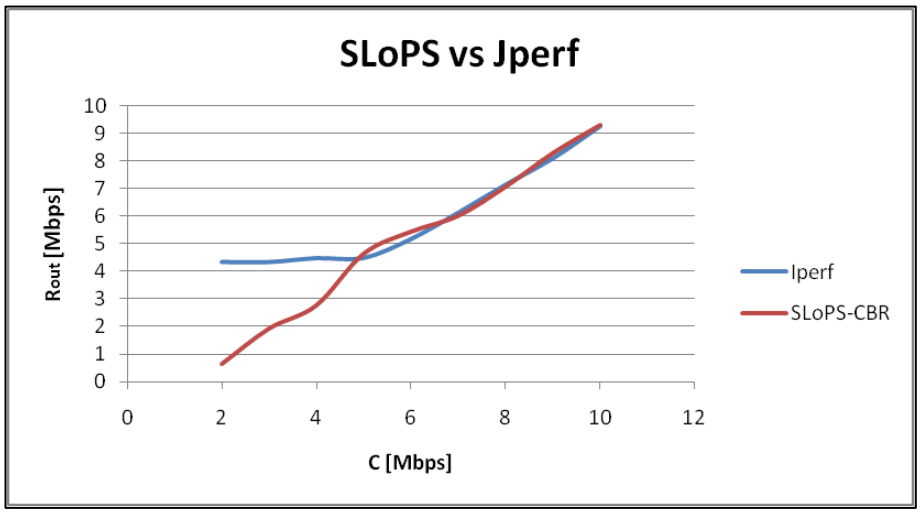

Figura 44. SLoPS con CBR

Para este algoritmo no se han realizado más pruebas, puesto que tiene pocos parámetros. Se completarán más adelante en el escenario final. La prueba realizada ha sido una pequeña demostración del funcionamiento del algoritmo en un escenario simple.

En este caso, el error en la zona en la que Iperf obtiene resultados similares a SLoPS, el error máximo cometido no supera el 5%.

5.1.3 Resultados de PathChirp

En este apartado se realizará alguna prueba relacionada con los parámetros para ver su funcionamiento. En la Figura 45 se muestra el resultado de *PathChirp* en función del tamaño del paquete utilizado (izqda.) y en función de sus parámetros F y L (dcha.) para poder obtener una estimación de cada uno de ellos.

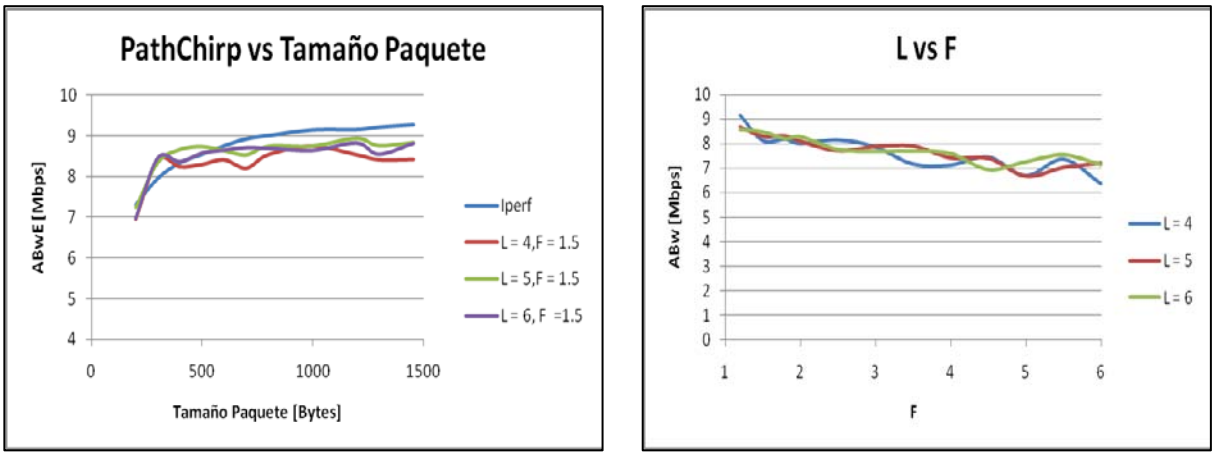

Figura 45. PathChirp en función del tamaño de paquete (izqda.) y L-F (dcha.)

Se puede comprobar que, para este algoritmo, el tamaño del paquete utilizado no afecta en gran medida aunque es aconsejable que esté entre 1000 y 1500 bytes. El error máximo cometido con respecto al Iperf es del 9,3 %.

En esta prueba, se puede observar que lo más recomendable es utilizar un valor de F entre 1 y 2.

5.1.4 Resultados de UDP

En este apartado se muestra el rendimiento del algoritmo más sencillo. En la Figura 46 se muestra dicho rendimiento y se compara con Iperf.

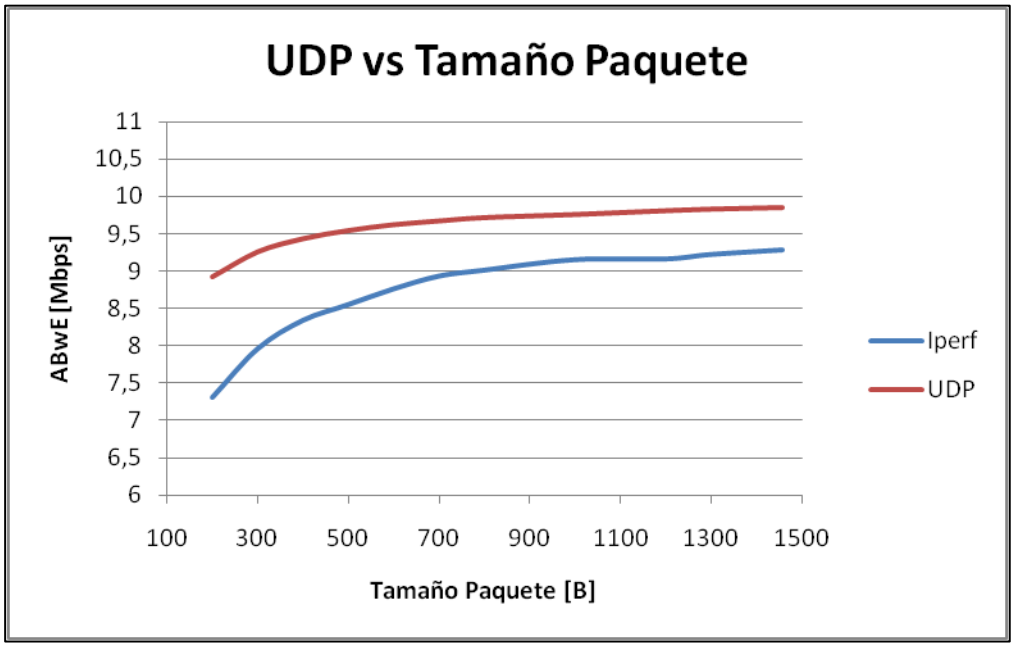

Figura 46. UDP en función del tamaño del paquete

Se puede observar cómo UDP obtiene resultados un poco superiores a los de Iperf pero mantiene la misma tendencia. Esto puede deberse a que con dicho algoritmo no se tienen en cuenta tiempos de transmisión ni retardos.

Por otro lado, al igual que en los algoritmos anteriores, se aprecia que el tamaño de paquete tiene que estar comprendido entre los 800 y 1500 bytes.

El error máximo cometido de UDP con respecto a Iperf es de un 22 % pero si sólo tenemos en cuenta la zona comprendida por el tamaño de paquete entre 800 y 1500 bytes, el error máximo es de un 9,2 %.

5.2 Escenario controlado generalizado usando el emulador WANem

Este escenario es fruto de un intento de probar la aplicación en un entorno controlado en el que se puedan obtener diferentes ABw conocidos, para poder obtener resultados distintos a los obtenidos anteriormente. De esta forma se podrán probar diferentes ABw y ver el comportamiento de los algoritmos.

Después de realizar un pequeño estudio sobre algunos emuladores de red se decidió utilizar uno llamado WANem [WEM]. La elección de dicho simulador fue debida a su fácil manejo e instalación aunque como se mostrará más adelante, se puede comprobar que los resultados obtenidos no han resultado especialmente buenos.

El apéndice 2 se dedica a la instalación y el uso de WANem.

En la Figura 47 se muestra el escenario en el que se realizaron las pruebas.

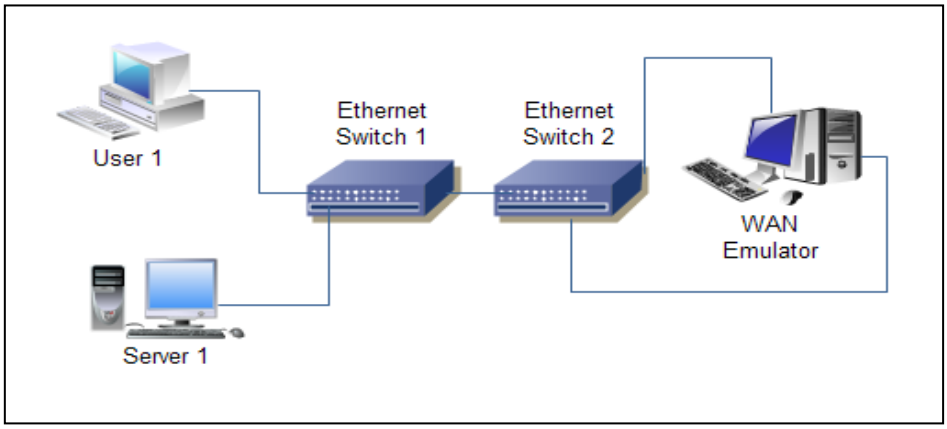

Figura 47. Escenario con el emulador WANem

El hardware empleado para este escenario se reflejan en la Tabla 17:

Parámetro	Servidor	Cliente	Emulador WANem
CPU	Intel Core 2 Duo 2100	Intel Core 2 Quad 2600	Pentium IV 2800
RAM	4 GB	4 GB	3 GB
Sistema Operativo	Linux Ubuntu 10.4	Windows 7	WANem
JVM	Versión 6.21 32 bits	Versión 6.19 64 bits	Versión 6.21 32 bits
Navegador	-	Mozilla y Chrome	-

Tabla 17. Hardware utilizado para el escenario con emulador

En cuanto a los resultados, cabe destacar varios aspectos importantes.

- El emulador funciona igual que un *Token Bucket*[4]. De esta manera, si la tasa que lo atraviesa no es mayor que la limitación impuesta, no detecta tal tasa y es como si no tuviera efecto. Por ejemplo, si se quiere que se limite a 20 Mbps, si no se envían paquetes en una tasa superior, ya sea en media o real, el limitador no se entera o no gasta los *tokens* y no actúa. Por esa razón, se introdujo en la aplicación

[4] Token Bucket: es un algoritmo de control de congestión basado en el conformado de tráfico. Es de tipo bucle abierto, lo que significa que previene la congestión (no reacciona cuando ya se ha producido, sino que previene que no se produzca), y lo hace conformando el tráfico que entra a la red para que ésta sea más determinista[TB].

el parámetro *Intrus*, con el que se muestra la tasa media [bits/seg] que se está enviando en cada prueba.

- Debido a la razón anterior expuesta, el algoritmo TOPP no se ha podido probar, puesto que es muy poco intrusivo y no se ha podido superar el límite impuesto.

- También ocurre con PathChirp cuando se quiere limitar con un ABw elevado. Además, Chirp no ofrece buenos resultados y lo más probable es que sea debido a que el limitador no es muy preciso.

- Con SLoPS pasa algo parecido que con PathChirp. A tasas bajas, funciona bien, pero a tasas elevadas no obtiene un buen resultado.

- Por el contrario, con UDP se detecta al ABw funciona a la perfección, puesto que es bastante intrusivo.

En la Figura 48 se muestran los resultados obtenidos con varias limitaciones de tasas para los tres algoritmos que funcionan, SLoPS, Chirp y UDP. Se puede comprobar, como a medida que aumenta el ABw, tanto SLoPS como Chirp van cometiendo más error.

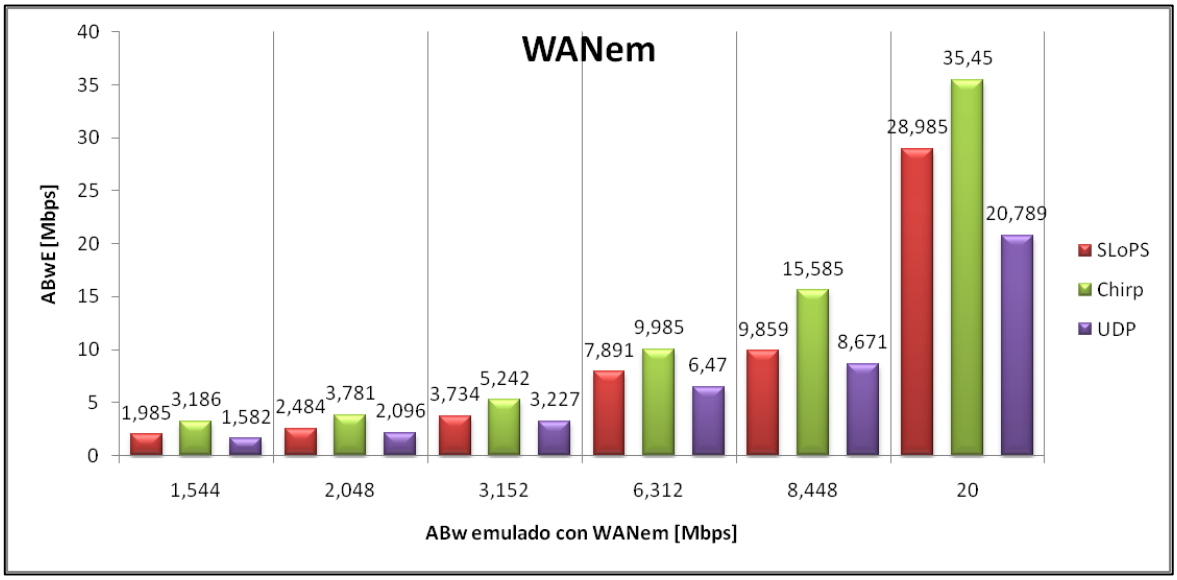

Figura 48. Resultados obtenidos utilizando WANem

El error cometido en cada uno de las pruebas se muestra en la Tabla 18.

ABw	1.544	2.048	3.152	6.312	8.448	20
SLoPS [%]	22,22	17,55	15,59	20,01	14,31	31,00
Chirp [%]	51,54	45,83	39,87	36,79	45,79	43,58
UDP [%]	2,40	2,29	2,32	2,44	2,57	3,80

Tabla 18. Errores cometidos con WANem

Se puede comprobar que el error cometido por UDP es bastante pequeño comparado con el cometido por el resto de algoritmos.

5.3 Escenario real

Este escenario es la prueba final de que la aplicación funciona y es donde realmente se puede realizar la verdadera comparación de cada uno de los algoritmos. Se comenzará mostrando los resultados obtenidos en dos entornos residenciales, uno con enlace de 3 y otro de 6 Mbps .

A continuación, se expondrán los resultados en un enlace de fibra óptica de 30 Mbps, donde se mostrarán además, otras pequeñas pruebas con las que se podrán justificar algunos valores de los parámetros establecidos. En este mismo apartado se mostrará un escenario donde se añadirá un enlace final a 10 Mbps para comprobar que también se detecta el cuello de botella aún estando en último lugar y en una red LAN después de atravesar Internet.

Finalmente se dedicará un escenario donde se expondrá un enlace con capacidad de 100 Mbps (Ethernet), denominado escenario de alta velocidad, donde se comprobarán sus limitaciones.

El hardware utilizado para este escenario se muestra en la Tabla 19

Parámetro	Servidor	Cliente
CPU	Intel Core	Intel Core 2 Duo 2100
RAM	4 GB	4 GB
Sistema Operativo	Linux	Windows 7
JVM	Versión 6.21 64 bits	Versión 6.19 64 bits
Navegador	-	Mozilla y Chrome

Tabla 19. Hardware utilizado para el escenario real

5.3.1 Escenario residencial con un enlace ADSL de 3 Mbps

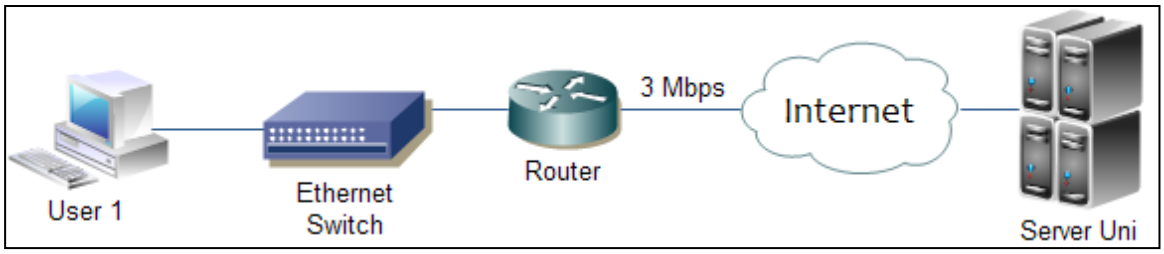

Figura 49. Escenario residencial con un enlace ADSL de 3 Mbps

En este escenario se ha podido comprobar el funcionamiento real de la aplicación. Se ha comprobado el ABw en el último enlace, el de 3 Mbps, que debería ser el más restrictivo. Para este caso, se muestran unos resultados finales comparando los algoritmos entre sí.

5.3.1.1 Resultados globales

En la Figura 50 se muestra una comparación de los resultados obtenidos entre los diferentes algoritmos expuestos. Además, se ha completado con dos métodos más para obtener así una validación:

- Test Ookla [Ook]: mencionado anteriormente en el capítulo 2.
- Test Movistar [TMST]: es un test de la pagina movistar que también puede valer para comparar.

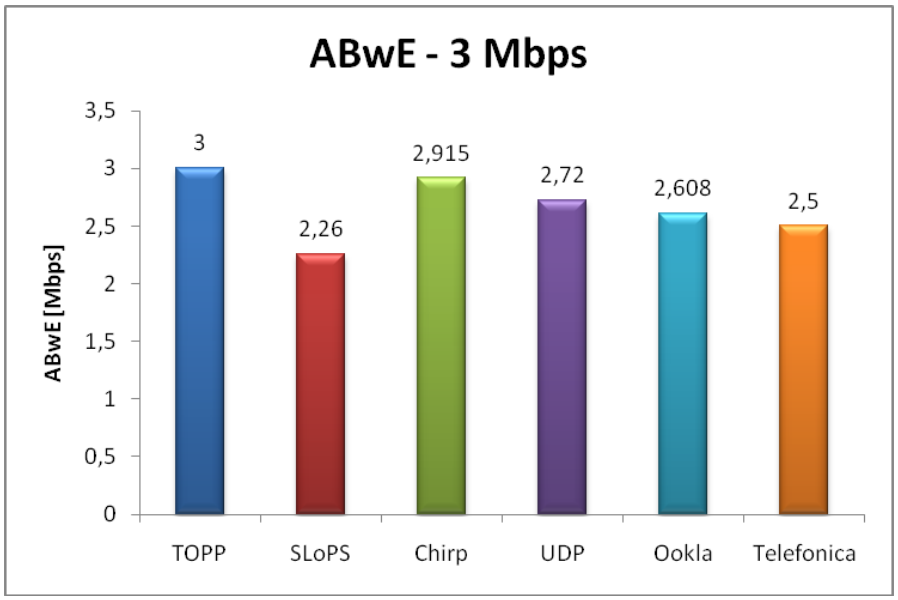

Figura 50. Resultados escenario residencial 3 Mbps

Se puede comprobar que todos los test obtienen resultados similares.

El error cometido sobre los dos algoritmos que no pertenecen a la aplicación se muestran en la Tabla 20:

Error [%]	TOPP	SLoPS	Chirp	UDP
Ookla	15,03	13,34	11,77	4,29
Telefónica	20,00	9,60	16,60	8,80

Tabla 20. Error cometido en el escenario completo a 3 Mbps

Se puede observar que el máximo error cometido por TOPP es de un 20 %.

5.3.2 Escenario residencial con un enlace ADSL de 6 Mbps

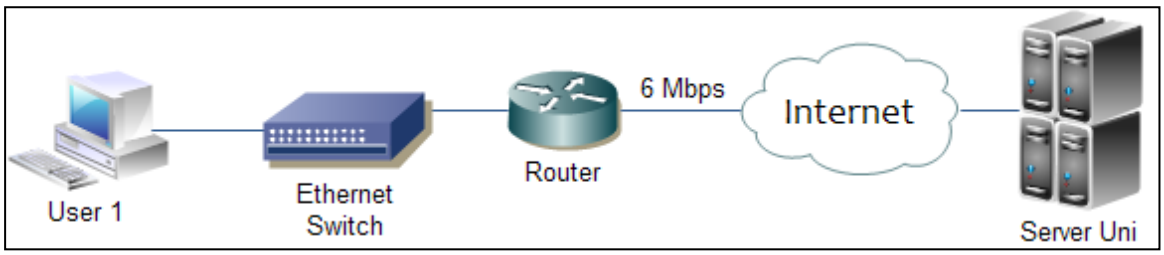

Figura 51. Escenario residencial con un enlace ADSL de 6 Mbps

Al igual que en el escenario anterior, también se ha querido comprobar el funcionamiento de la aplicación cuando el enlace es de 6 Mbps.

5.3.2.1 Resultados globales

En este caso, la Figura 52 (izqda.) muestra los resultados obtenidos.

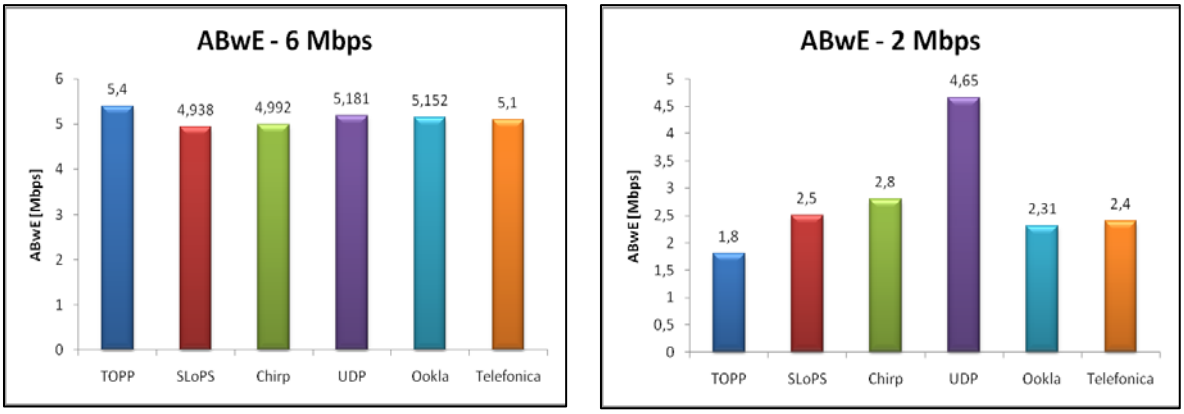

Figura 52. Resultados escenario residencial 6 Mbps con (dcha.) y sin descarga (izqda.)

Se puede comprobar que los resultados son prácticamente similares para cada uno de los algoritmos. Además se realiza la validación con los dos métodos convencionales ya mencionados en el escenario anterior. Se puede apreciar, que el ancho de banda ofrecido de 6 Mbps no se cumple al 100 %.

El error cometido en estas pruebas corresponde a los valores de la Tabla 21.

Error [%]	TOPP	SLoPS	Chirp	UDP
Ookla	4,81	4,15	3,11	0,56
Telefónica	5,88	3,18	2,12	1,59

Tabla 21. Error escenario residencial a 6 Mbps

También se ha querido probar dicha medida mientras se descarga algún fichero de la red. Véase Figura 52 (dcha.).

Esta prueba se ha realizado mientras se descargaba un fichero a una tasa de 4 Mbps (500 KB/s). Esto se muestra en la Figura 52 (dcha.). Durante el test realizado se pudo comprobar que, salvo el Ookla y el UDP, la medida no afectaba a dicha tasa. En el caso de Ookla y UDP, se observó como la tasa disminuía (400 KB/s) para adaptarse al nuevo tráfico. Esto permite comprobar la intrusión de los algoritmos. Aún así, Ookla presenta un resultado bastante coherente.

El error cometido por cada uno de ellos con respecto a los métodos externos a la aplicación se muestra en la Tabla 22.

Error [%]	TOPP	SLoPS	Chirp	UDP
Ookla	22,08	8,23	21,21	101,30
Telefónica	25,00	4,17	16,67	93,75

Tabla 22. Error a 6 Mbps con descarga

En este caso se puede observar como UDP muestra un elevado error. Realmente no es un error tal y como se ha comentado, sino que debido a la intrusión tan elevada, ha conseguido obtener ese ancho de banda para poder enviar los paquetes.

5.3.3 Escenario residencial con un enlace de fibra de 30 Mbps

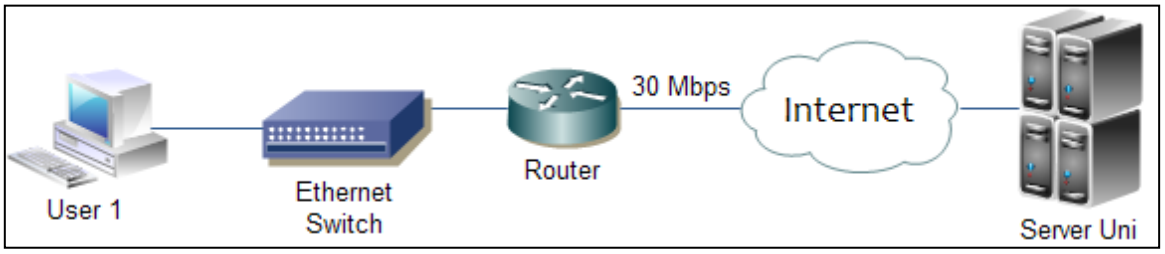

Figura 53. Escenario residencial con un enlace de fibra de 30 Mbps

Completado un poco el escenario anterior, con este otro escenario se ha verificado el funcionamiento de la aplicación con un ABw mayor.

Además de los resultados finales, se van a mostrar diferentes pruebas realizadas con cada uno de los algoritmos son el fin de probar un poco más su rendimiento y poder obtener unos parámetros óptimos en cada algoritmo.

En este apartado se podrá comprobar el funcionamiento de los algoritmos con diferentes configuraciones, de forma que se podrán obtener valores óptimos para algunos de los parámetros de cada algoritmo. Esto no quiere decir que sean siempre óptimos. En este caso serán óptimos en el momento de realizar las pruebas, pero las condiciones Internet son tan cambiantes hoy en día que puede que en otro momento estos valores no sean óptimos.

5.3.3.1 Resultados globales

En este apartado, se muestra unos resultados globales en los que se puede comprobar el funcionamiento general de la aplicación.

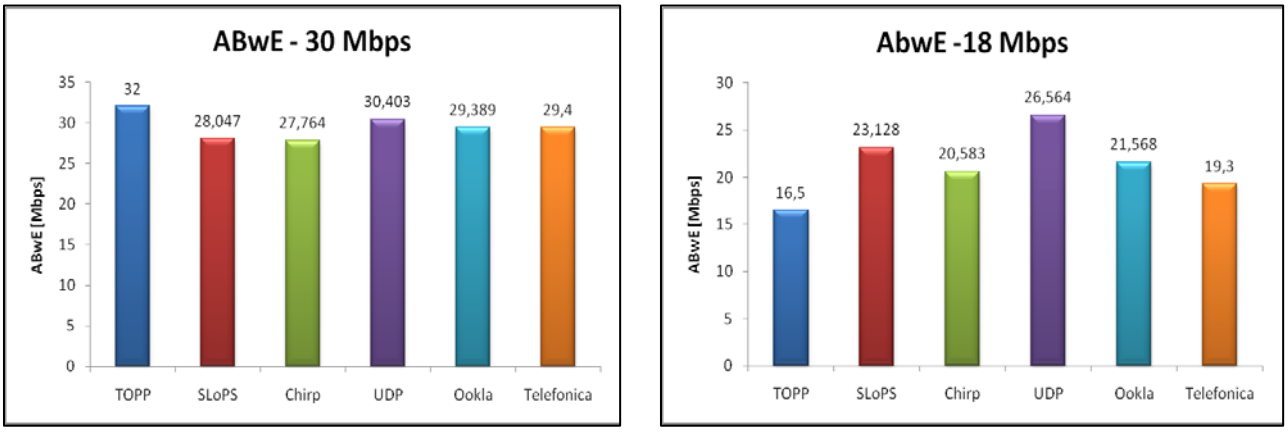

Figura 54. Resultados escenario residencial 30 Mbps con (dcha.) y sin descarga (izqa.)

En la Figura 54 se muestra una comparación de los resultados obtenidos entre los diferentes algoritmos expuestos. Se puede comprobar que todos los test obtienen resultados similares.

En la Tabla 23 se muestran el error que cada uno de ellos comete con respecto a los dos métodos externos a la aplicación.

Error [%]	TOPP	SLoPS	Chirp	UDP
Ookla	8,88	4,57	5,53	3,45
Telefónica	8,84	4,60	5,56	3,41

Tabla 23. Error escenario residencial a 30 Mbps

Al igual que en los escenarios anteriores, también se ha probado estimar el ABw cuando se está descargando un fichero desde la red. Concretamente se estaba descargando a una tasa de 12 Mbps (aprox.). En la Figura 54 (dcha.) se muestran dichos resultados y en la Tabla 24 se muestran el error correspondiente a cada uno de ellos.

Error [%]	TOPP	SLoPS	Chirp	UDP
Ookla	23,50	7,23	4,57	23,16
Telefónica	14,51	19,83	6,65	37,64

Tabla 24. Error con descarga

5.3.3.1.1 Resultados TOPP

En este apartado se realizarán algunas pruebas sobre el algoritmo TOPP con el fin de ajustar sus parámetros de forma que su rendimiento sea lo más óptimo posible.

En primer lugar, se mostrará el rendimiento que obtenido en función del tamaño de los paquetes. Esto se puede ver en la Figura 55.

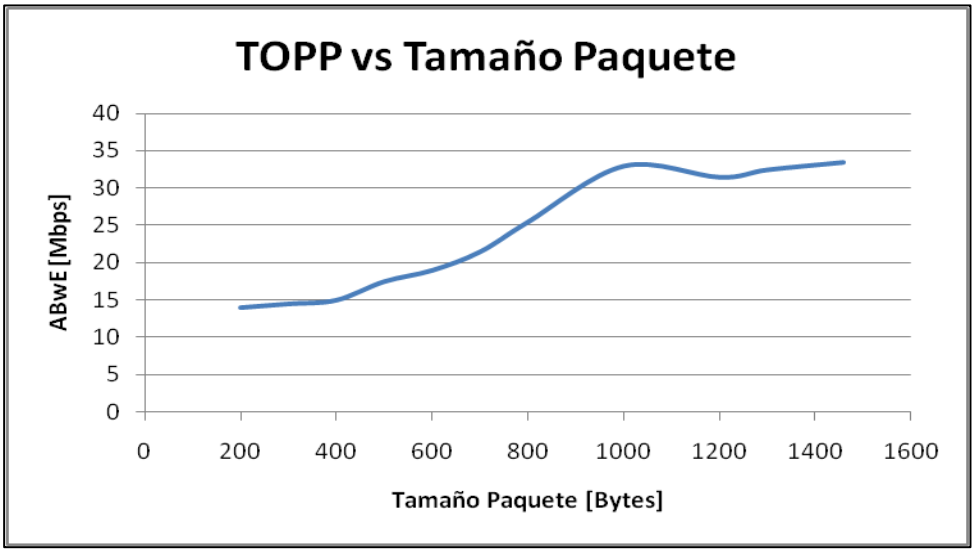

Figura 55. TOPP en función del tamaño de los paquetes

Se puede comprobar que utilizando valores entre 1000 y 1500 bytes el resultado obtenido se ajusta más al ABw real y, como se mencionó en anteriores secciones, cuanto mayor sea mejor. De este modo no se encontrará con tráfico cruzado con un tamaño mayor que el suyo en muchas de las redes por donde pase. Este resultado está también sesgado por el rendimiento de Java y de Ethernet, puesto que como se ha visto en apartados anteriores, el tamaño del paquete es un parámetro importante cuando se utiliza el protocolo UDP para este tipo de medidas.

Una segunda medida realizada, ha sido el rendimiento obtenido en función del número de paquetes K utilizado. En la Figura 56 se muestra el resultado en el que se puede ver que afecta muy poco, aunque se puede observar que un valor óptimo para K puede ser entre 8 y 14 paquetes.

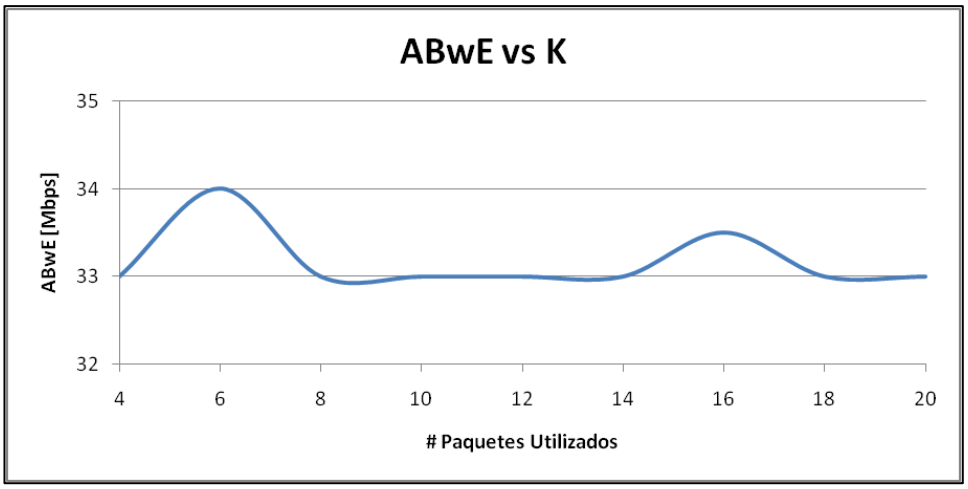

Figura 56. Rendimiento TOPP en función de K utilizados

Como tercera medida, se decidió obtener el rendimiento en función del parámetro $T_{in}ME$ que es, como ya se ha mencionado anteriormente, el margen de tiempo que se establece en la diferencia de tiempos entre paquetes. En la Figura 57 se puede observar que el margen ideal está en entre un 8% y un 12%, puesto que son los valores en el que los resultados son más estables y están más cercanos a su medida real.

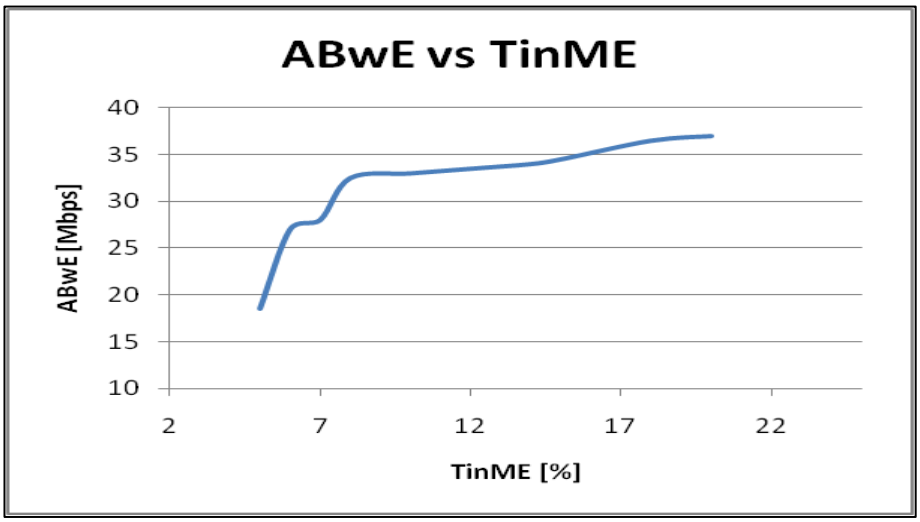

Figura 57. Rendimiento TOPP en función de $T_{in}ME$

Por último, en la Figura 58 se mostrará el rendimiento obtenido en función del número de pares de paquetes que llegan dentro del margen establecido (RPOK). El $T_{in}ME$ establecido para esta prueba ha sido de un 10%. El resultado muestra claramente que cuantos menos paquetes se dejen pasar en ese margen, se obtendrá un resultado más restrictivo. Por ello, el valor óptimo de este parámetro estará comprendido entre el 50% y 60 %.

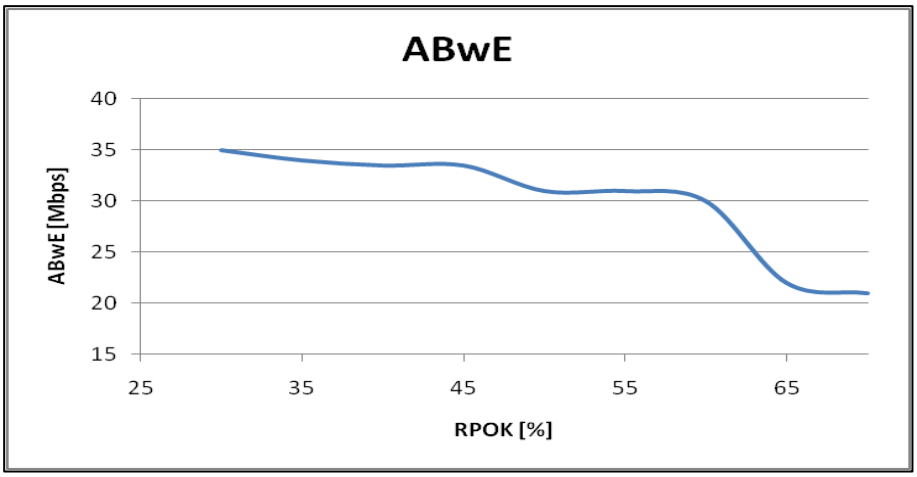

Figura 58. Rendimiento TOPP frente RPOK

5.3.3.1.2 Resultados SLoPS

Para SLoPS se han realizado las siguientes pruebas para comprobar su rendimiento.

En primer lugar, se ha medido su rendimiento en función del tamaño máximo de paquete. Este parámetro está limitado por el rango de tasas de envío puesto que el algoritmo funciona de forma que va cambiando su tamaño de paquete a medida que va variando la tasa a probar. De esta forma, cuanto más pequeño sea el tamaño máximo habrá menos variación en el tamaño de los paquetes, pero aún así, al igual que en TOPP, cuanto mayor sea el tamaño de paquete utilizado, mejores resultados se obtienen. Esto se muestra en la Figura 59.

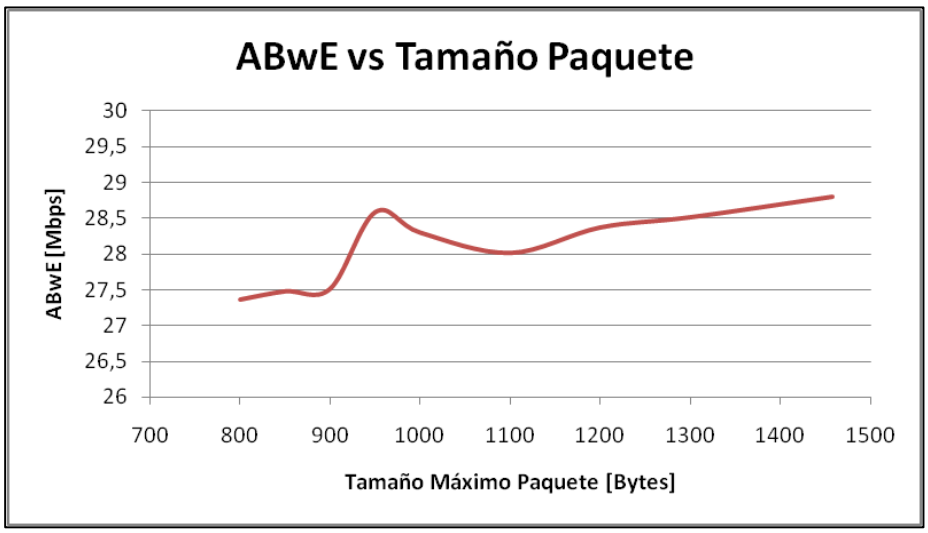

Figura 59. Rendimiento SLoPS frente al tamaño máximo de paquete

En segundo lugar, se mostrará el rendimiento en función del número de paquetes utilizado para la medida. En la Figura 60 se puede observar que este parámetro no afecta en gran medida al resultado final, pero hay que llegar a un compromiso entre el número de paquetes y la intrusismo del algoritmo, es decir, cuanto mayor sea el número de paquetes utilizado mejor será la medida, puesto que hay un mejor análisis estadístico, pero también ocurre que existe un mayor número de paquetes en la red y, por tanto, el algoritmo se vuelve más intrusivo y más lento.

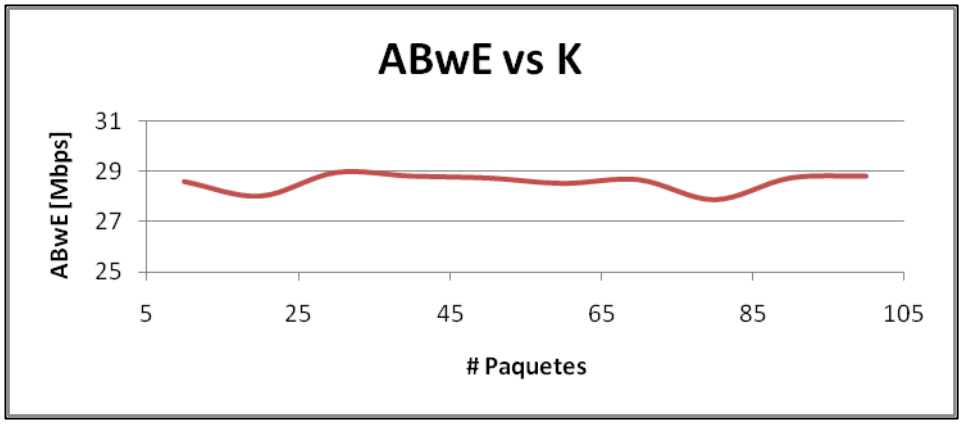

Figura 60. Rendimiento SLoPS en función del número de paquetes

A la vista de los resultados, un valor que cumpla el compromiso podría estar entre 25 y 65 paquetes por flujo.

Una tercera prueba es comprobar el rendimiento en función de los parámetros S_{pct} y S_{pdt}. En la Figura 61 podemos ver como estos parámetros toman especial importancia.

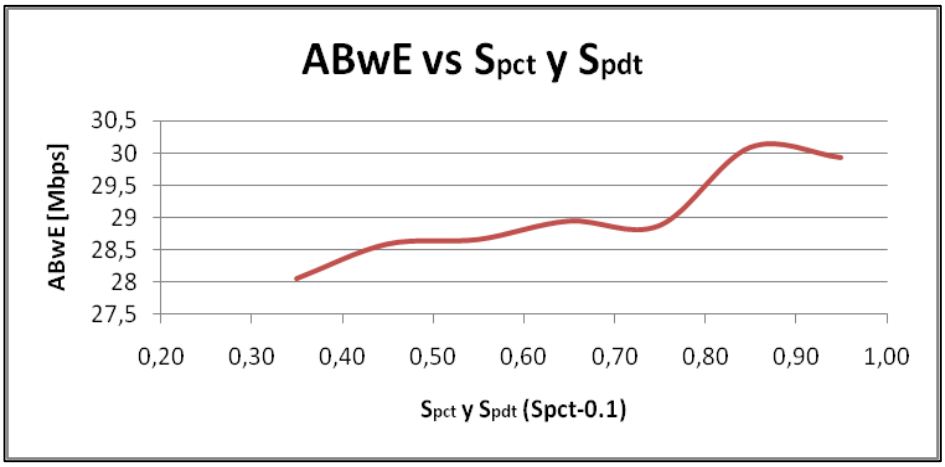

Figura 61. Rendimiento SLoPS en función de S_{pct} y S_{pdt}

Aunque la variación no es demasiado grande, estos parámetros no conviene ni que sean muy bajos ni muy altos. Si son muy bajos, el algoritmo detectará congestión un mayor número de veces e incluso puede que detecte congestión cuando todavía no exista realmente. Si son muy altos, ocurrirá lo contrario, puede que no se detecte congestión

cuando realmente esté existiendo. Valores óptimos para estos parámetros pueden ser entre 0,66 - 0,76 para el S_{pct} y entre 0,56 - 0,66 para S_{pdt}.

Para terminar, el último parámetro que se ha probado es el umbral f con el que se determina si en una ráfaga ha habido una tendencia creciente o decreciente en función del análisis estadístico obtenido con los parámetros probados anteriormente. En la Figura 62 se muestra el rendimiento en función de dicho parámetro:

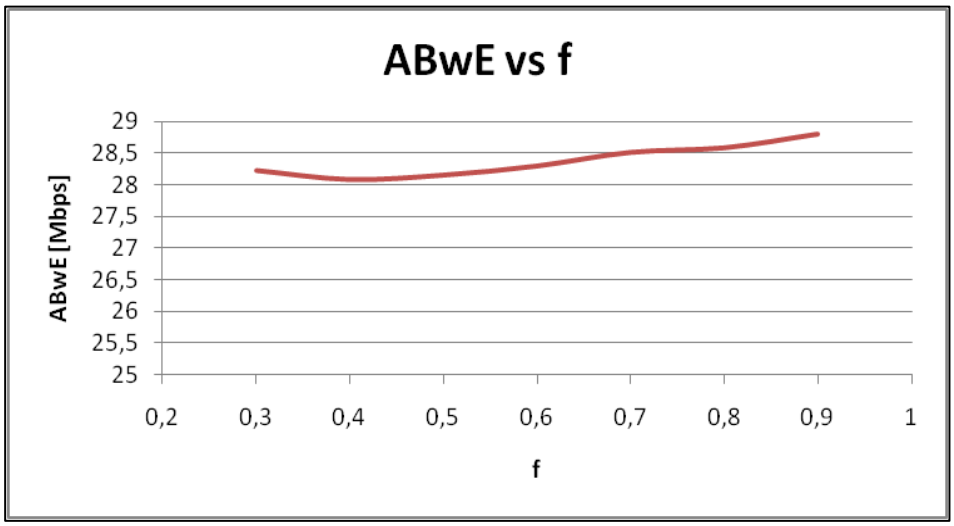

Figura 62. Rendimiento SLoPS en función del umbral f

Se puede ver, al igual que con los parámetros S_{pct} y S_{pdt}, que la variación no es demasiado notable; de la misma forma, un valor muy bajo puede no determinar bien si se ha producido una ráfaga con tendencia creciente o decreciente. Un valor óptimo para este parámetro podría ser 0,7.

5.3.3.1.3 Resultados PathChirp

PathChirp tiene menos parámetros que los anteriores. Además es un algoritmo más sencillo, mas rápido y poco intrusivo. Como ya hemos visto en anteriores resultados, es el algoritmo que mejor se adapta a la red y el que mejor rendimiento ofrece.

Con PathChirp se han realizado las siguientes pruebas para ver su rendimiento:

La primera, al igual que con todos los anteriores, es el rendimiento en función del tamaño de los paquetes utilizados. En la Figura 63 se muestra el resultado de dichas pruebas. En ella, se puede comprobar que el tamaño óptimo de los paquetes ha de estar comprendido entre 800 y 1500 Bytes.

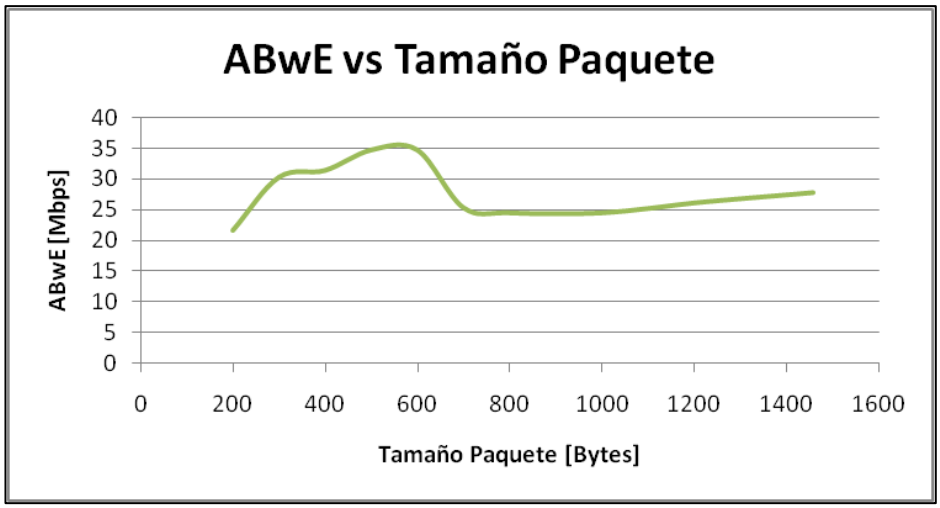

Figura 63. Rendimiento PathChirp en función del tamaño de los paquetes

Se puede apreciar, que con paquetes menores de dicho tamaño el algoritmo parece inestable. Este comportamiento puede deberse a que con tamaños pequeños, el algoritmo no detecta la mayor congestión existente en ese momento.

Una segunda medida, es obtener los valores de L y F, aunque, como se ha mencionado en capítulos anteriores, estos valores son muy difíciles de obtener. Realizando una pequeña gráfica, se puede obtener una idea de cuál puede ser un valor razonable.

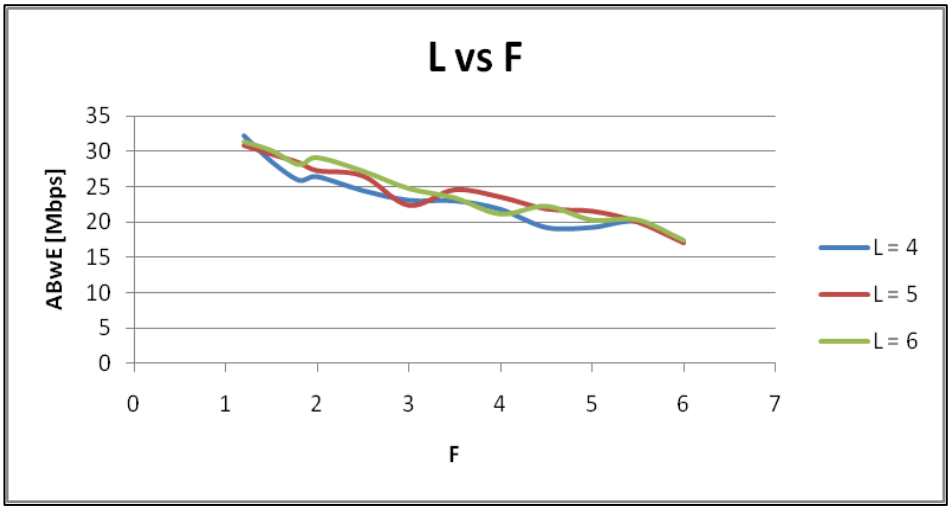

Figura 64. Rendimiento PathChirp en función de L y F

En la Figura 64 podemos ver como un valor óptimo para F puede estar comprendido entre 1 y 2, mientras que L no es un factor muy determinante. Lo mejor es que este entre 4 y 6, puesto que es el umbral con el que se determina si una excursión es válida o no.

5.3.3.1.4 Resultados UDP

Para este algoritmo, puesto que es tan simple, solo se ha medido el rendimiento en función del tamaño del paquete. Esto se muestra en la Figura 65.

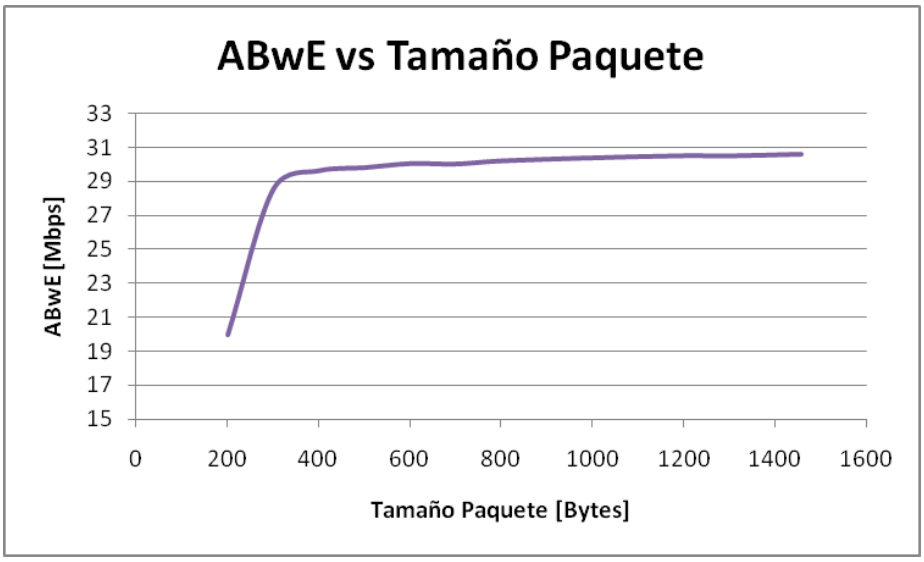

Figura 65. Rendimiento UDP en función del tamaño del paquete

Se puede comprobar, que el tamaño del paquete no afecta salvo para tamaños muy pequeños, aunque es recomendable utilizar siempre tamaños de paquete entre 800 y 1500 bytes.

5.3.3.2 Escenario residencial con un enlace de fibra de 30 Mbps y capacidad 10 Mbps en el último enlace

En el escenario mostrado en la Figura 66, tan sólo se quiere comprobar cómo afecta el resultado añadiendo un salto más. Concretamente un último salto de 10 Mbps en el último enlace. Como se verá en los resultados de la Figura 67, los resultados obtenidos son coherentes y no existe ningún efecto extraño en la medida final.

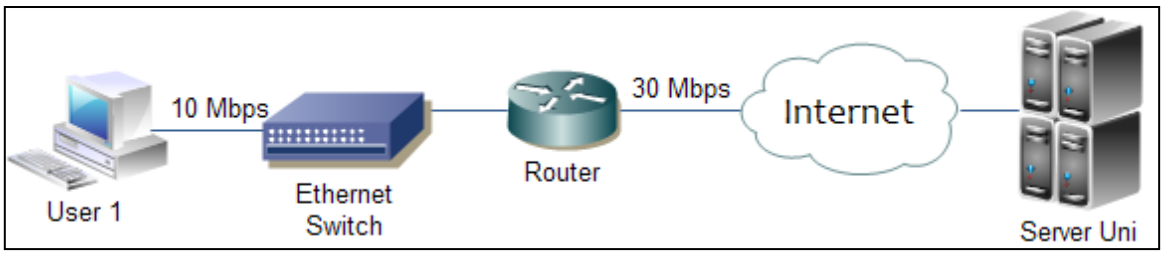

Figura 66. Escenario residencial con un último enlace a 10 Mbps

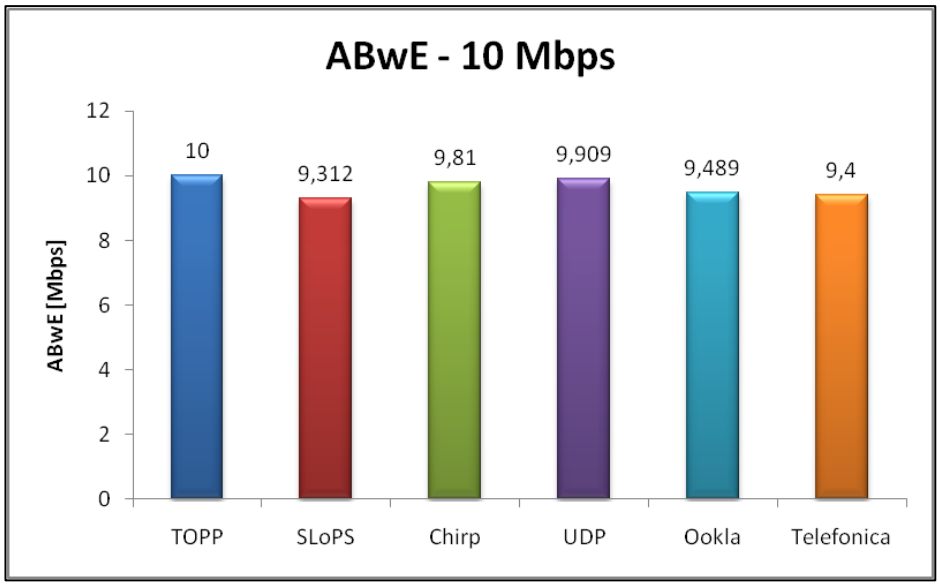

Figura 67. Resultados escenario residencial a 10 Mbps

Se puede comprobar, que todos los algoritmos obtienen resultados similares. Además, también se ha realizado la prueba con los dos métodos adicionales ya utilizados en apartados anteriores y el resultado es bastante aproximado.

Los errores cometidos con respecto al cada uno de los algoritmos no pertenecientes a la aplicación se muestran en la Tabla 25:

Error [%]	TOPP	SLoPS	Chirp	UDP
Ookla	5,39	1,87	3,38	4,43
Telefónica	6,38	0,94	4,36	5,41

Tabla 25. Errores con respecto Ookla y Telefónica

5.4 Escenario a alta velocidad

Este escenario es el más simple de todos. Se trata de probarlo en la red Ethernet sin ningún tipo de congestión. En la Figura 68 se muestra el escenarios, que es muy parecido al escenario simple, salvo sin el tráfico cruzado y sin el enlace de 10 Mbps.

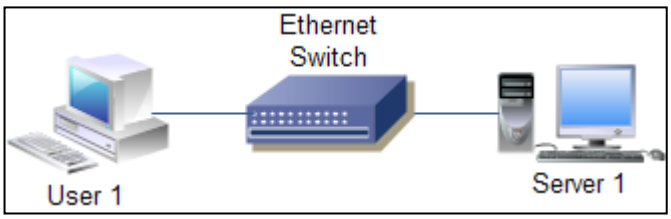

Figura 68. Escenario de alta velocidad

Los resultados obtenidos en este escenario se muestran en la Figura 69.

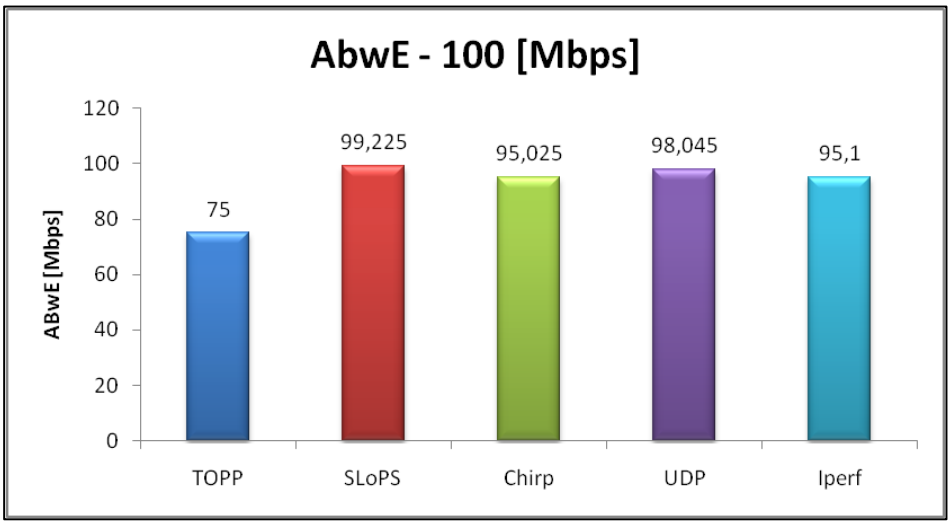

Figura 69. Resultados escenario alta velocidad

Se puede comprobar que TOPP es el que peor resultado obtiene. Esto se debe a todos los problemas que se han ido mencionando en apartados anteriores. El resto de algoritmos obtienen mejores resultados y se aproximan mejor al resultado mostrado con Iperf, con él que, en este caso, podemos realizar la validación.

Los errores obtenidos con cada uno de los algoritmos con respecto a Iperf de muestran en la Tabla 26:

	TOPP	SLoPS	Chirp	UDP
Error [%]	21,13	4,33	0,08	3,10

Tabla 26. Error con respecto a Iperf

Además, se ha realizado una segunda medida en la que podemos observar el comportamiento de UDP e Iperf con respecto al tamaño del paquete. Esto se pude ver en la Figura 70.

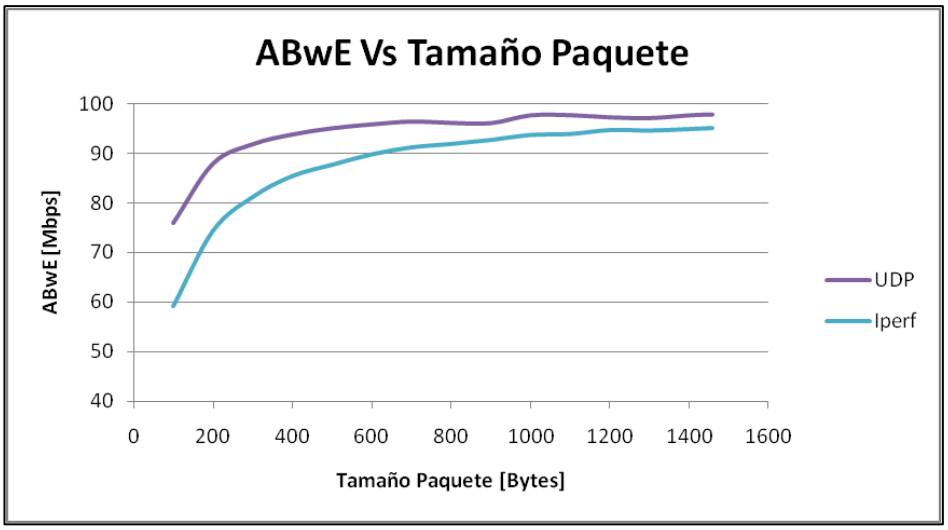

Figura 70. UDP vs Iperf con respecto al tamaño del paquete

Una vez más podemos ver como el tamaño del paquete es fundamental. En este caso también se sigue cumpliendo que el valor óptimo para el tamaño del paquete está entre 800 y 1500 bytes, donde el máximo error que se comete con UDP es de un 4,4 %.

Capítulo 6

Problemas encontrados

En este capítulo se describen algunos de los problemas encontrados durante el desarrollo del proyecto. Como ya se verá, están relacionados con la capacidad que tiene Java para manejar las medidas realizadas y la obtención de valores precisos para dicha mediadas.

6.1 Problemas de rendimiento

Aunque la ventaja que posee Java con respecto a la multiplataforma es muy conveniente para muchas aplicaciones, en este caso no ha dado su mejor rendimiento. Para poder tomar medidas de los tiempos, Java depende de la Máquina virtual (JVM), que a su vez, es la que está haciendo de pasarela con el sistema operativo (SO). Esto hace que el rendimiento, a la hora de realizar las medidas sea menor, puesto que la obtención de los tiempos se toma a nivel de aplicación. Esto se hace así porque Java no posee herramientas que accedan a la medición a un nivel más bajo.

De esta forma se ha comprobado que, en algunos momentos, los paquetes que llegan desde el emisor, con una separación establecida, son detectados por la aplicación con otra separación diferente, lo que estropea la medida. Esto se debe a que los recursos de los que dispone Java en ese momento no son los óptimos para la medida y no recibe los paquetes

en el tiempo que realmente han llegado. Este efecto se agrava en mayor medida cuando los paquetes llegan con tiempos más cercanos entre sí, es decir, con tasas elevadas.

Para demostrar esto, se ha diseñado el escenario mostrado en la Figura 71.

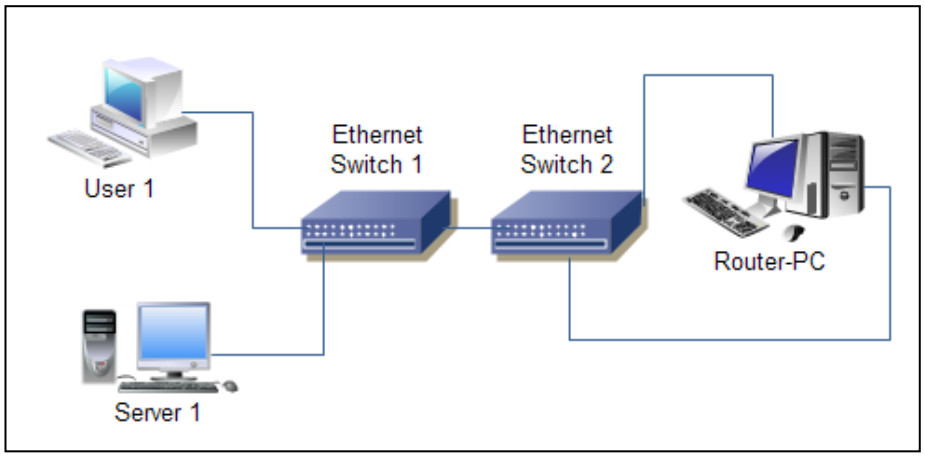

Figura 71. Escenario para medir fallos de rendimiento

Una vez realizadas las pruebas en dicho escenario, se pudo comprobar que los paquetes que pasaban a través del PC-Router, en el que se observan cómo entran y salen, atravesaban la red correctamente, pero luego, en el cliente, no se detectaba lo mismo. Esto quiere decir que, de alguna forma, Java afecta en la medida. De esta manera, lo que puede ocurrir es que el SO no ofrezca los recursos necesarios a Java para obtener una medida precisa de los paquetes cuando llegan, de manera que Java los obtienen encolados o de alguna otra manera distinta a la que realmente han viajado por la red, con lo que la medida resultante es totalmente errónea.

A continuación, se mostrarán un par de ejemplos en los que se muestra la situación descrita.

En primer lugar, en la Figura 72 se muestra una situación de buen funcionamiento. En dicha figura, se muestran el tiempo entre paquetes que mide Java (azul) y los que mide el Router-PC (rojo). Podemos ver que los dos miden prácticamente lo mismo. Se puede ver como los paquetes empiezan con una distancia entre ellos en torno a unos 150 useg.

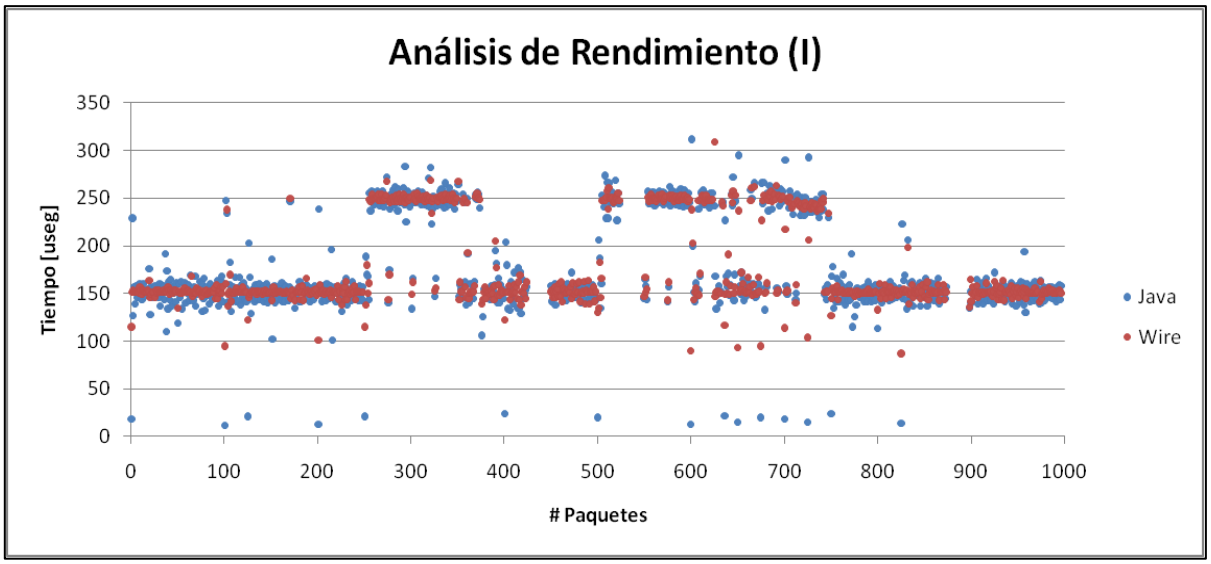

Figura 72. Comparación entre Java y Router-PC

En cuanto se detecta congestión, dicha separación aumenta, lo que provocará un incremento en el OWD y luego se ven franjas en las que no se vuelve a encontrar congestión.

El OWD se puede ver en la Figura 73, en donde vemos los OWD de un envió a tasa sin congestión y un envío a una tasa con congestión.

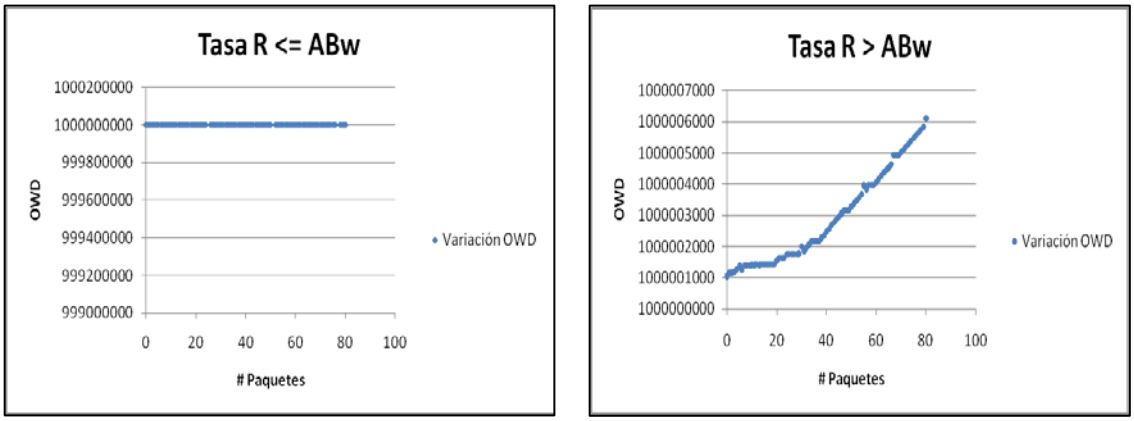

Figura 73. OWD no creciente (izqda.) y OWD creciente (dcha.)

En la Figura 74 se muestra un ejemplo de mal funcionamiento. Se puede apreciar como los paquetes que están en azul, Java los mide mal ya que los paquetes llegan con tres separaciones claramente diferenciadas, mientras que los paquetes que están en rojo muestran un buen comportamiento. Esto afectará directamente a TOPP, puesto que mide de forma directa dicho tiempo.

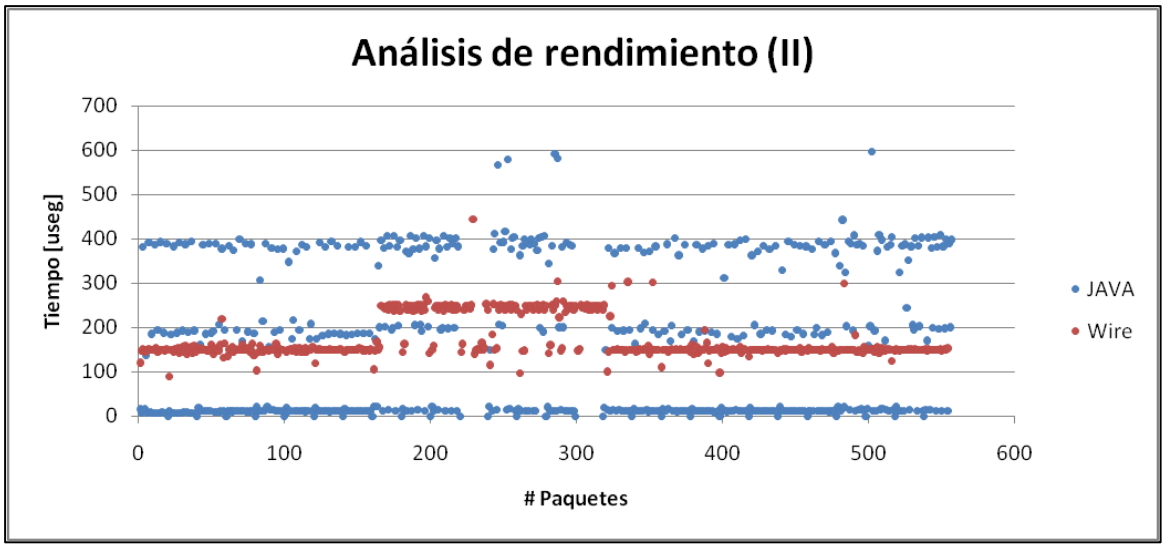

Figura 74. Java y Router-PC caso irregular

Sin embargo, a la hora de obtener el OWD se ha observado que dicho efecto no se aprecia de igual forma, con lo que SLoPS y *PathChirp* no se verán tan afectados por este efecto. En la Figura 75 se puede apreciar que durante el periodo creciente, el OWD crece. Esto es así, porque la medida recogida del OWD es claramente creciente, aunque la distancia entre los paquetes no sea la establecida.

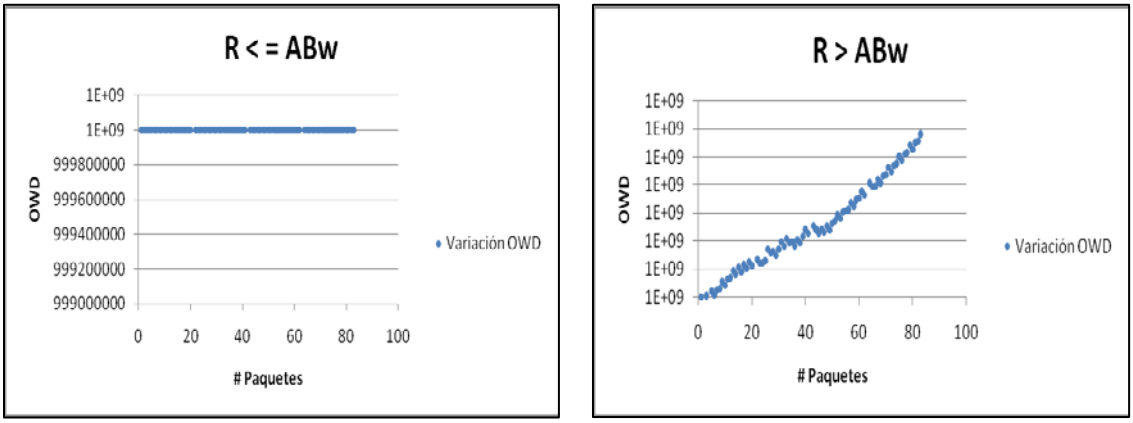

Figura 75. OWD no creciente (izqda.) y OWD creciente (dcha.) con detección errónea

A la vista de estos resultados se puede decir que el OWD es una forma bastante robusta de obtener la medida, pero también se han visto casos en los que la situación anterior fallaba, con lo que se puede concluir que Java no es el mejor lenguaje de programación para obtener medidas de este tipo. Por otro lado, si las tasas a medir no son elevadas (entre 1 - 60 Mbps), posiblemente el efecto de la falta de recursos no se aprecia tanto y Java puede ser utilizado perfectamente.

6.2 Sincronización

Aunque en capítulos anteriores se haya mencionado que la sincronización no es necesaria para medir el OWD, sí que es importante que exista un mínimo de sincronización en cuanto a que el emisor tiene que tomar los tiempos sobre un punto de referencia menor que en el receptor. De no ser así, a la hora de calcular y operar con el OWD hay que tener en cuenta que los valores negativos hay que tratarlos de forma diferente.

Cuando se calcula la variación del OWD, si tenemos valores negativos, cuando existe retardo, éste se hace más negativo, con lo que hay que tomarlo como si creciese aunque matemáticamente indica que decrece. El valor absoluto de esto no funciona puesto que precisamente pasa eso, que se tomaría como si decreciera y no es verdad. En Java es tremendamente complicado sincronizar en tiempo dos PCs utilizando la función System.nanoTime().

La solución más sencilla es poner un offset para que los tiempos tomados estén correctamente dispuestos, es decir, $t_r > t_s$ en cualquier caso.

Ejemplo:

En Java, la función System.nanoTime() nos da un tiempo en nanosegundos tomado desde un punto de referencia arbitrario. De esta forma, en cada máquina, y en cada sesión, será diferente. La solución más sencilla encontrada es la siguiente:

si $t_r = 7192999999$ y $t_s = 9213131313$, caso problemático, se puede corregir poniendo los dos tiempos en la parte media del rango. Es decir $t_r = 500000000$ y $t_s = 400000000$.

Estos offsets se calculan con los primeros valores y luego se aplica al resto. De esta forma, en ningún momento existirá error. Este offset no afecta a la medida ya que siempre se utilizan datos relativos, con lo que los posibles offset se cancelan y además como son siempre los mismos, no afectan para medir incrementos del OWD, que es lo que utilizamos.

Capítulo 7

Presupuesto

7.1 Programación del proyecto

El proyecto se dividió en cinco fases. A continuación se indican las etapas de cada fase para poder presentar la programación de *gantt* del proyecto.

7.1.1 Estudio previo

Tarea 1: estudio y selección de las metodologías más adecuadas. Se consultaron todas las fuentes disponibles en busca de la documentación necesaria para realizar un primer proceso de estudio y consideración de soluciones. El tiempo empleado fue de un mes.

Tarea 2: selección de los algoritmos más relevantes. En esta fase se incluye el estudio de toda la documentación localizada previamente. El objetivo fue la obtención de una clara idea de las tecnologías y metodologías existentes para estimar el ABw. De esta forma se abordó el resto de las etapas del proyecto. Esta tarea llevó un tiempo de un mes.

Tarea 3: primera aplicación simple. En esta fase se realizaron pequeñas comprobaciones para determinar si Java era idóneo para realizar el proyecto. Se implementó una pequeña

aplicación en la que se probaron algunas de las características de los algoritmos. Una vez comprobada la aplicación se procedió a desarrollar la aplicación final en paralelo con la memoria del proyecto correspondiente al estudio previo y el desarrollo de la aplicación. Esta fase duró un mes.

7.1.2 Implementación

Tarea 4: el primer paso fue seleccionar e instalar un entorno de desarrollo cómodo y sencillo para este tipo de aplicaciones. De entre un gran número de entornos, el escogido fue Eclipse. Las razones fueron que además de ser gratuito, incorporaba plugins bastante útiles y sencillos para desarrollar la parte grafica, esta parte duró cuatro días.

Tarea 5: como segundo paso, se buscó información en la API de Java sobre como implementar la aplicación. En esta parte se buscó la mejor forma de poder implementar el cliente y el servidor. Se realizaron pequeñas pruebas y se compararon distintos métodos para implementarlo. Esta fase duró ocho días.

Tarea 6: diseño de las clases y la arquitectura. Se realizó un primer esquema de clases Java para poder implementar la aplicación. En base a este primer esquema, se implementaron los algoritmos. Esta fase se realizó en dos días.

Tarea 7: implementación del primer algoritmo. En esta fase se implementó el esquema general del primer algoritmo, de forma que después se pudiera incorporar a la interfaz gráfica, es decir, siendo un bloque independiente. Esta fase duró dieciocho días.

Tarea 8: implementación del segundo algoritmo. Este algoritmo costó un poco más implementarlo debido a que tenía una complejidad un poco mayor que él anterior y aunque ya se tenía una base con el algoritmo anterior, esta fase llevó veintidós días.
Implementación de una primera versión de la interfaz grafica. Se desarrolló interfaz grafica para poder ir ensamblando los diferentes algoritmos

Tarea 9: implementación del tercer algoritmo. Este algoritmo es el que menos costó, puesto que ya se tenía una base y además el algoritmo es más sencillo. Duró diez días.

Tarea 10: diseño de la arquitectura TCP. Esta fase se centró en diseñar la manera de establecer la conexión entre el cliente y el servidor y como establecer un diálogo entre los dos para realizar los test. Se tardó dos días.

Tarea 11: implementación de la interfaz gráfica. Aunque ya se tenía una base de dicha interfaz, se realizaron grandes modificaciones y su desarrollo fue bastante complejo. Esta tarea duró un mes.

Durante seis días se realizaron pequeñas pruebas para ver si todo estaba correcto, pero se detectaron fallos y se realzó un estudio más exhaustivo para determinar los fallos.

Tarea 12: búsqueda de fallos. Esta tarea es una de las que más tiempo llevó, puesto que fue difícil encontrar los fallos. Se invirtieron dos meses.

7.1.3 Depuración

Tarea 13: diseño del banco de pruebas. Una vez desarrollado el software, se realizó un plan de pruebas para validar el proyecto. En esta fase se configuró un banco de pruebas con distintos escenarios para poder mostrar los resultados. Dichos escenarios se pensaron de forma que se pudieran visualizar diferentes alternativas para poder observar el rendimiento de la aplicación. Esta labor llevó cuatro días.

Tarea 14: realización de las pruebas en los escenarios propuestos. Esta fase duró un mes y diez días en las que, debido a la gran cantidad de medidas que eran necesarias y el posterior reporte de las modificaciones pertinentes al diseño, hizo que fuese una fase de tan larga duración. Después de obtener los resultados, se documentaron de tal forma que posteriormente se pudiera elaborar la documentación correspondiente para la entrega final del proyecto.

7.1.4 Entrega

Tarea 15: elaboración de memoria del proyecto. Esta fase llevó dos meses, se trató de plasmar en un documento todos los puntos que constituían el proyecto. Se inició esta tarea al finalizar. Esta fase comenzó después de realizar el estudio previo desarrollado durante las fases uno y dos. Se dejó cerrado el contexto teórico y se comenzó a desarrollar la aplicación. Por ello, esta fase solo corresponde a la realización de los primeros capítulos y se tardó 20 días en completar.

Tarea 16: elaboración de los capítulos correspondiente a la aplicación y a los resultados. En esta fase, se plasmaron todos los desarrollos y resultados obtenidos. La elaboración de dichos capítulos llevó dos meses y 20 días.

Tarea 17: elaboración del resto de capítulos. En esta fase se completaron los capítulos restantes tales como conclusiones, glosario, apéndices, etc.. Esta fase llevó un mes y 10 días.

Una vez realizada la memoria, se presentó el proyecto. Para ello se prepararon algunos materiales adicionales, lo que llevó cinco días incluyendo la presentación.

7.2 Diagrama de gantt

En la Figura 76 se puede apreciar cada una de las fases del proyecto que se marcó como un hito en la programación, comprobando que el tiempo total que llevó completar el proyecto fue de 8 meses puesto que ha habido periodos de inactividad.

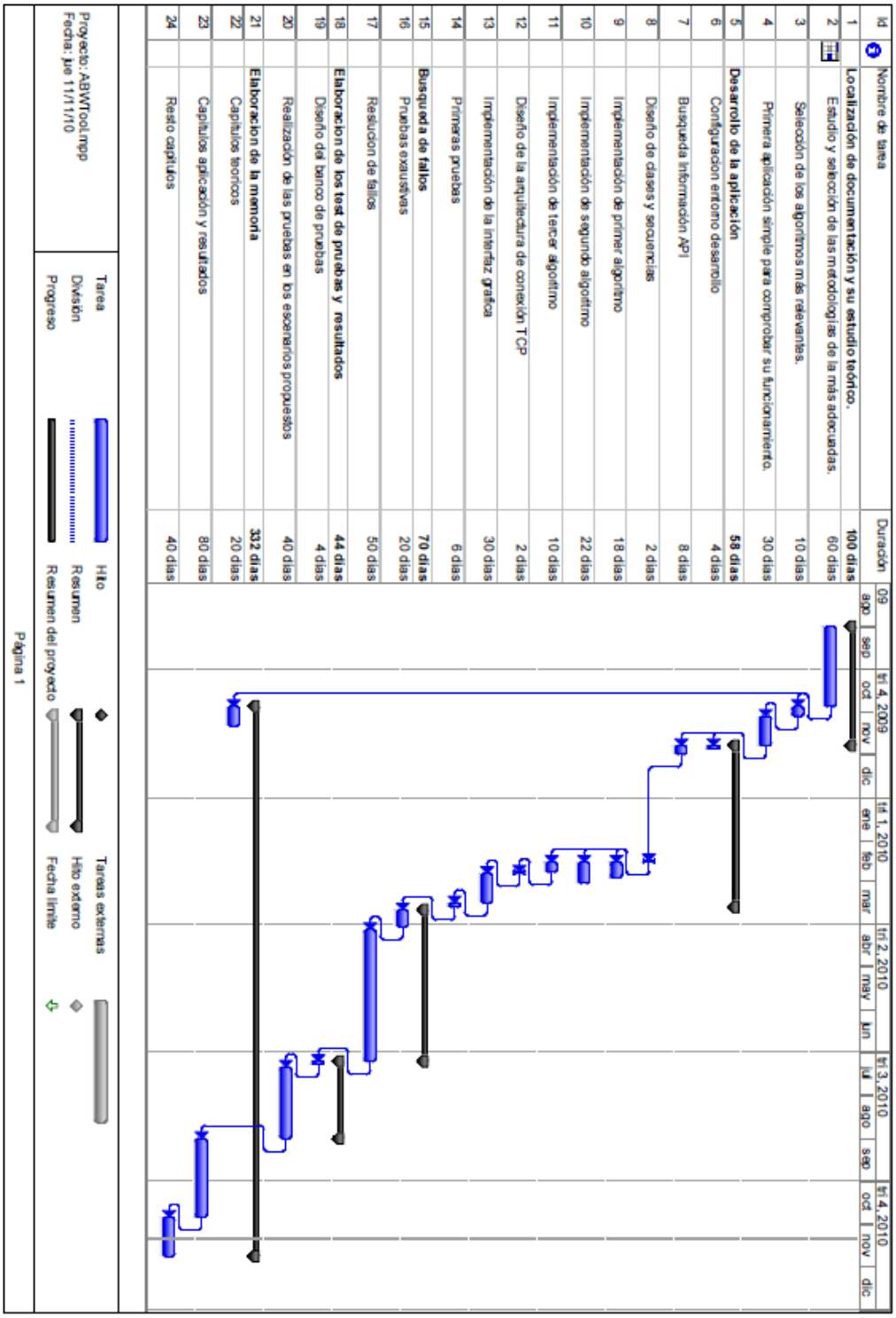

Figura 76 Diagrama de *gantt* del proyecto

7.3 Análisis de costes

Para este apartado se utilizará la plantilla propuesta por la Universidad Carlos III de Madrid [UC3M].

PRESUPUESTO DE PROYECTO

1.- Autor:

Manuel Antolín Ayuso

2.- Departamento:

Departamento de Ingeniería Telemática

3.- Descripción del Proyecto:

- Titulo	**ESTUDIO DE LA ESTIMACIÓN DEL ANCHO DE BANDA DISPONIBLE EN COMUNICACIONES EXTREMO A EXTREMO**
- Duración (meses)	**8**
Tasa de costes Indirectos:	**20%**

4.- Presupuesto total del Proyecto (valores en Euros):

26.042 Euros

5.- Desglose presupuestario (costes directos)

PERSONAL

Apellidos y nombre	N.I.F. (no rellenar - solo a título informativo)	Categoría	Dedicación (hombres mes) [a)]	Coste hombre mes	Coste (Euro)
					0,00
			0	4.289,54	0,00
Antolín Ayuso, Manuel		Ingeniero	8	2.694,39	21.555,12
					0,00
					0,00
				Total	21.555,12

[a)] 1 Hombre mes = 131,25 horas. Máximo anual de dedicación de 12 hombres mes (1575 horas)
Máximo anual para PDI de la Universidad Carlos III de Madrid de 8,8 hombres mes (1.155 horas)

EQUIPOS

Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable [d]
Equipo PC Sobremesa	600,00	100	8	60	80,00
Equipo PC Portátil	500,00	100	6	60	50,00
Dos Equipos PC Auxiliares	200,00	100	4	60	13,33
Dos tarjetas de Red extra	50,00	100	4	60	3,33
		100		60	0,00
					0,00
				Total	146,67

[d] Fórmula de cálculo de la Amortización:

$$\frac{A}{B} x C x D$$

A = nº de meses desde la fecha de facturación en que el equipo es utilizado.
B = periodo de depreciación (60 meses).
C = coste del equipo (sin IVA)
D = % del uso que se dedica al proyecto (habitualmente 100%).

SUBCONTRATACIÓN DE TAREAS

Descripción	Empresa	Coste imputable
	Total	0,00

OTROS COSTES DIRECTOS DEL PROYECTO [e]

Descripción	Empresa	Costes imputable
	Total	0,00

[e] Este capítulo de gastos incluye todos los gastos no contemplados en los conceptos anteriores, por ejemplo: fungible, viajes y dietas, otros,…

6.- Resumen de costes

Presupuesto Costes Totales	**Presupuesto Costes Totales**
Personal	21.555
Amortización	147
Subcontratación de tareas	0
Costes de funcionamiento	0
Costes Indirectos	4.340
Total	26.042

El presupuesto total de este proyecto asciende a la cantidad de VEINTISEIS MIL CUARENTA Y DOS EUROS.

Leganés 15 de diciembre de 2010,

El ingeniero proyectista

Fdo. Manuel Antolín Ayuso

Capítulo 8

Conclusiones y trabajo futuro

8.1 Conclusiones

El objetivo final del proyecto es diseñar una aplicación que permita comparar diferentes métodos para medir el ABw extremo a extremo. Además se ha completado con un pequeño estudio sobre los algoritmos implementados con el que se explican sus principales particularidades.

A la vista de los resultados obtenidos, la principal conclusión resultante es que los algoritmos estiman el ABw de forma válida. Estos algoritmos tienen la ventaja de que son menos intrusivos y más simples que los usuales. El algoritmo que mejores prestaciones ofrece es claramente *PathChirp*, puesto que es bastante rápido, poco intrusivo y muy sencillo de implementar. Con TOPP se han observado varios problemas en distintos aspectos como por ejemplo en lo que le afecta el tráfico cruzado o en la precisión de la medida, por lo que no es de los más adecuados, y aunque es muy poco intrusivo, necesita muchas iteraciones y mucho tiempo para llegar a un resultado. SLoPS es bastante intrusivo, pero efectivo aunque también necesita bastante tiempo, puesto que el numero de iteraciones no es fijo y no se sabe a priori, para realizar la medida.

En la mayoría de las pruebas realizadas se ha comprobado que los aspectos teóricos mencionados en el capítulo 3 se cumplen, aunque el uso de Java ha añadido alguna problemática más. Dicha problemática plantea el problema que tiene Java en cuanto a la obtención de la medida. Debido en gran parte a lo que se ha explicado en el Capítulo 6 sobre los recursos de Java, no es posible obtener una medida precisa de tiempo entre dos paquetes consecutivos cuando se está midiendo anchos de banda elevados. Como las medidas se toman cada menos tiempo, los recursos necesarios son mayores y el SO no los ofrece el tiempo necesario para realizar la medida de una forma precisa. Este problema de recursos repercute directamente en los algoritmos de forma que:

- Esta limitación influye sobre todo en las mediciones del ABw cercanos a 60 Mbps o superiores y de esa forma no es posible realizar adecuadamente la medida Esto afecta más directamente al algoritmo TOPP. En los otros algoritmos, se ha visto que funciona correctamente en la mayoría de los casos, pero también se han observado irregularidades en determinados momentos.

- En el momento en el que los recursos no son los óptimos, para el caso de TOPP, la medida puede verse afectada a cualquier tasa, pero en especial a las tasas elevadas.

- Sin embargo, como se ha podido comprobar, SLoPS y PathChirp funcionan mejor puesto que utilizan el OWD para obtener la medida y se ha comprobado que es más robusto.

Otra conclusión importante a destacar es que, dependiendo del uso que se dé a estos algoritmos para obtener el ABw, el resultado puede no ser beneficioso para el usuario. Debido a que el resultado que se obtiene es el ABw, quiere decir que si la red está ocupada, en teoría no se podría acceder a la red o no podría hacerlo a una tasa lo suficientemente buena. Si se está utilizando estos algoritmos para que otra aplicación utilice la red de forma que solo utilice el ABw en ese momento, puede ocurrir que no pueda empezar nunca o que lo haga a tasas muy bajas por que se basaría en un resultado puntual y esperaría a que hubiera más ABw. Esto vulnera el principio de compartición de la red, en el que todos los usuarios tienen derecho a poder usar la red en las mismas condiciones.

En cambio, si se utilizan los medios convencionales mediante TCP, debido a que controla el flujo, todos los usuarios se adaptan a la red, y si entra una conexión nueva, se ajustarían todos por igual. De esta manera, no se recomendaría el uso de estos algoritmos para este tipo de aplicaciones.

El uso principal de estos algoritmos podría ser para poder obtener el ABw de una red y poder medir sus prestaciones sin necesidad de que la medida influyera de forma notable en la red.

Como conclusión final se puede asegurar que la medida del ABw, cuando está comprendido entre 1 y 30 Mbps, es válida. Además, como ya se ha mencionado, no es demasiado intrusiva, por lo que no afecta demasiado a la red y pueden ser unos buenos métodos para empezar a desarrollar. Sobre todo PathChirp, que es el que mejores prestaciones ofrece.

8.2 Trabajo Futuro

Como trabajo futuro se proponen proponer las siguientes acciones:

- Mejorar la aplicación gráficamente, se pueden añadir gráficos de forma que se vea como van llegando los paquetes (aunque no sea en tiempo real).

- Completar con alguna herramienta de obtención de estadísticas.

- Añadir medidas de retardos

- Añadir algún otro algoritmo que se quiera comparar.

- Añadir una opción para medir el ABw up-stream, opción no contemplada por no ser parte del objetivo principal del proyecto.

- También se puede utilizar Java Real-Time o JNI (Java Native Interface), aunque ya no se podría aprovechar la ventaja multiplataforma que ofrece Java, pero se mejoraría notablemente el rendimiento de los algoritmos.

- Mejora de los algoritmos, es posible que se puedan optimizar, sobre todo PathChirp, en el que ya hay estudios sobre las posibles mejoras para optimizar su rendimiento y poder implementar una aplicación, tal como son las que existen hoy en día basadas en la transferencia de ficheros, que pueda obtener el ancho de banda disponible de una forma más eficiente.

- Implementación de una versión reducida de esta aplicación para redes móviles y poder medir el ABw en una red móvil mediante cualquier terminal. En este caso, Java no sería una limitación, puesto que por el momento, las redes móviles no tienen una tasa de transmisión elevada.

- Diseño de un protocolo específico para utilizar los algoritmos directamente entre routers. De esa forma, cada router podrá saber la congestión existente en sus enlaces y podrá seleccionar el mejor camino posible.

Capítulo 9

Bibliografía

9.1 Documetos Publicados

[Cas+06] Castellanos, C.U.,Villa, D.L.,Teyeb, O.M.,Elling, J.,Wigard, J.: Comparison of Available Bandwidth Estimation Techniques in Packet-Switched Mobile Networks, 2006.

[CM02] Liang Cheng and Ivan Marsic, Java-based tools for accurate bandwidth measurement of Digital Subscriber Line networks, 2002.

[CV06] Castellanos, C.U.,Villa, D.L. Study of Available Bandwidth Estimation Techniques to be applied in Packet-Switched Mobile Networks, May 2006.

[DRM01] Constantinos Dovrolis, Parameswaran Ramanathan, and David Moore: What do packet dispersion techniques measure?,2001.

[HS03] Ningning Hu and Peter Steenkiste. Evaluation and characterization of available bandwidth probing techniques, 2003.

[JC02] Manish Jain and Constantinos Dovrolis. Pathload: A measurement tool for end-to-end available bandwidth, 2002.

[JD03] Manish Jain and Constantinos Dovrolis. End-to-end available bandwidth: measurement methodology, dynamics, and relation with TCP throughput, 2003.

[JMB04] Andreas Johnsson, Bob Melander, and Mats Björkman. DietTopp: A first implementation and evaluation of a simplified bandwidth measurement method. In Second Swedish National Computer Networking Workshop, page 5, Karlstad, November 2004.

[Kap+04] Rohit Kapoor, Ling-Jyh Chen, Li Lao, Mario Gerla, M. Y. Sanadidi, CapProbe: A Simple and Accurate Capacity Estimation Technique, 2004.

[LDS06] Lao, L., Dovrolis, C., Sanadidi, M.Y.: The probe gap model can underestimate the available bandwidth of multihop paths, 2006.

[MBG00] Bob Melander, Mats Björkman, and Per Gunningberg. A new end-toend probing and analysis method for estimating bandwidth bottlenecks. IEEE Global Internet Symposium, November 2000.

[Pra+03] R.Prasad, C.Dovrolis, M.Murray and K.Claffy. Bandwidth estimation: Metrics, measurements techniques, and tools. IEEE Network, November 2003.

[Rib+00] Vinay J. Ribeiro, Mark Coates, Rudolf H. Riedi, Shriram Sarvotham, Brent Hendricks, and Richard Baraniuk. Multifractal cross-traffic estimation. In Proc. of ITC Specialist Seminar on IP Traffic Measurement, September 2000.

[Rib+03] Vinay J. Ribeiro, Rudolf H. Riedi, Richard G. Baraniuk, Jiri Navratil, and Les Cottrell. pathChirp: Effcient available bandwidth estimation for network paths. In Passive and Active Measurement Workshop, April 2003.

[SKK03] J. Strauss, D. Katabi, and F. Kaashoek. A measurement study of available band width estimation tools. In Proceedings of Internet Measurement Conference, October 2003.

9.2 Request For Coments

[IANA] http://www.iana.org/assignments/port-numbers, última visita 15/11/2010.

[RFC-0768] RFC-0768 UDP http://www.faqs.org/rfcs/rfc768.html, última visita 15/11/2010.

[RFC-0793] RFC-0793 TCP, http://www.rfc-es.org/rfc/rfc0793-es.txt, última visita 15/11/2010.

9.3 Páginas Web

[Ook] Test Ookla: http://www.testsdevelocidad.es/, última visita 15/11/2010.

[UC3M] Presupuesto UC3M, https://www.uc3m.es/portal/page/portal/administracion_campus_leganes_est_cg/proyecto_fin_carrera, última visita 15/11/2010.

[Ram08] Ramón, Franciso Javier: http://www.lacofa.es/index.php/tecnologias/tests-de-velocidad-para-banda-ancha-i-la-importancia-de-una-estimacion-precisa, última visita 15/11/2010.

[TB] Token Bucket, http://www.it.uc3m.es/~prometeo/rsc/problemas/Examenes_uc3m/feb98/teoria~1/teoria~1.html, última visita 15/11/2010.

[TMST] Test Movistar: http://www.movistar.es/on/io/es/micro/test_velocidad/test_velocidad_telefonica.htm, última visita 15/11/2010.

9.4 Software

[DITG] Generador de tráfico D-ITG, http://www.grid.unina.it/software/ITG/, última visita 15/11/2010.

[ECS] Eclipse, entrono de desarrollo Java, http://www.eclipse.org/, última visita 15/11/2010.

[END] Edraw Network Diagram, http://www.edrawsoft.com/, última visita 15/11/2010.

[JDK] Java Development Kit, http://www.oracle.com/technetwork/java/javase/downloads/index.html, última visita 15/11/2010.

[JRE] Java Runtime Edition, http://www.java.com/es/, última visita 15/11/2010.

[JPF] Jperf, sofware de medición de ancho de banda, http://iperf.sourceforge.net/, http://sourceforge.net/projects/jperf/, última visita 15/11/2010.

[TDP] TcpDump, Network Protocol Analyzer.

[UBT] Sistema Operativo Linux Ubuntu, http://www.ubuntu.com/ , última visita 15/11/2010.

[WEM] WANem, emulador de redes, http://wanem.sourceforge.net/, última visita 15/11/2010.

[WIN7] Sistema Operativo Microsoft Windows 7, http://emea.microsoftstore.com/es/es-ES/Microsoft/Windows/Windows-7, última visita 15/11/2010.

[WSK] WireShark, Network Protocol Analyzer, http://www.wireshark.org, última visita 15/11/2010.

Apéndice 1: ABMTool User´s Guide

This appendix will show a quick guide to use ABMTool.

1.1 Introduction

Available Bandwidth Measurement Tool is a complete tool in which you can get the ABw in many different ways. The application is based on an applet and there is no need for installation. When the applet is running, you can test the ABw with several algorithm.

In this guide, it will be explained the different ways to run the algorithms, with their optimal value parameters and also it could be seen how to the application shows the results.

1.2 Connection

The first dashboard is about the connection. It is showed in the next picture.

This dashboard configures and sets the TCP and UDP connection. The applet is connected to the server via TCP and each algorithm is connected via UDP.

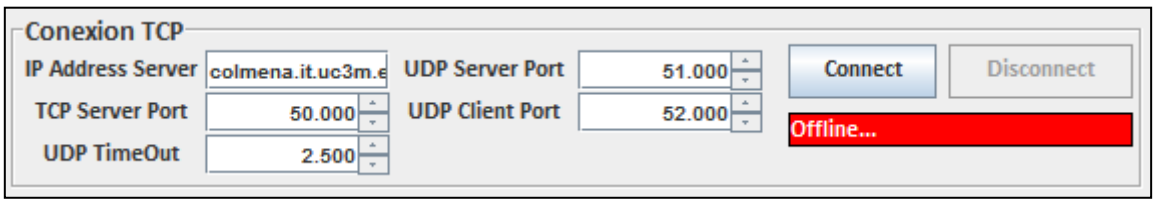

Figure 77. Connection dashboard

Through this dashboard, it could be configured the following parameters:

Server IP Address: it is the server IP address or the URL where the application server is running.

TCP Server Port: the TCP port where the server is listening. The default value is 50000, because it is the port where the server is listening. If the connection is not established, it could be because the server is not running or is busy due to another user.

Server Port UDP: the UDP port in which the server UDP is listening. By default, the value of this port is 51000 although it could be changed. As higher the port number is, it would be easier to find a free port.

Client UDP Port: the UDP Port in which the client is listening. The default value is 52000. You can choose whatever you want because it is your host. The default value could be the desired option.

UDP Time Out: it is the maximum time limit waiting for a client to receive packages. It is important that this value isn´t very high because it makes the test too long. But if necessary it could be risen up. The default value is 2,5 seconds.

In addition, there is a status bar indicating the connection status. The values of the states are as follows:

Offline (red): the connection status is offline. There is no connection between client and server. It could be that the server is not running or it might be busy.

Connecting (yellow): the connection is being established.

Connected (green): the connection is set.

1.3 TOPP

Parameters

The Figure 78 shows the TOPP panel with the corresponding structure, as is required to send packages. In this structure some of the most important parameters are shown.

TOPP is based in Packet Pairs Techniques. As shown on the right picture on each iteration, the parameter T_{in} is changed in order to test a differnt rate.

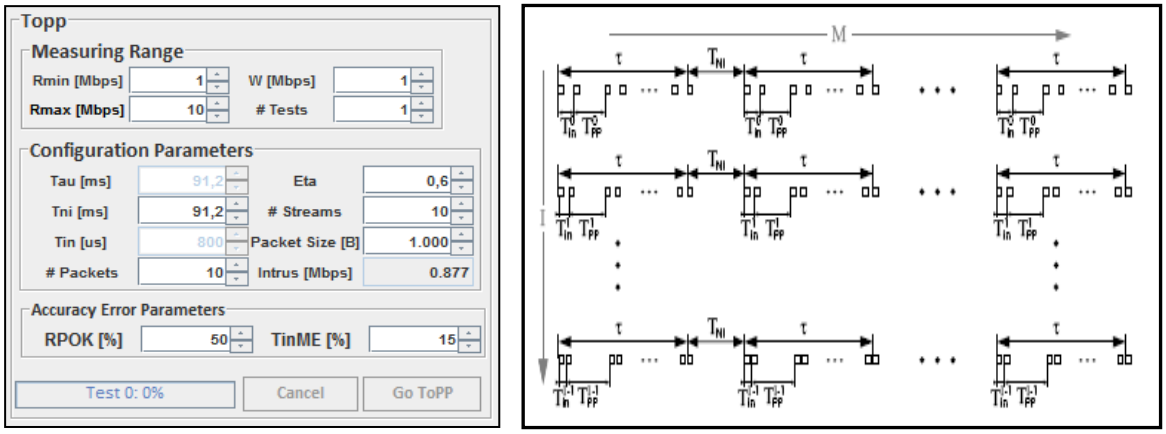

Figure 78. TOPP configuration

On top of the TOPP panel, a measuring range sub-panel it is shown. There are four parameters:

R_{min}: it is the minimum rate in Mbps to test. This value is very important, because if you set a higher value than the real bandwidth, the algorithm shows zero. But if you set a low value the test time will be too long. The optimal values could be:

if you think that your bandwidth is about 6 Mbps, you must set R_{min} at 1 or 2 Mbps. On the other hand, if you think your bandwidth is 30 Mbps, it should be set R_{min} at 10 or 15 Mbps.

R_{max}: it is the maximum rate in Mbps to test. If you set a high value it is possible that the algorithm tries more tests than necessary. The optimal values could be:

if you think your bandwidth is close to 6 Mbps, you must set R_{max} at 8 or 10 Mbps. On the other hand, if you believe your bandwidth is 30 Mbps, it should be set R_{max} at 35 or 40 Mbps.

W: it represents the resolution in Mbps. In this case it is the step of each iteration. The smaller value, the better results will be got. But on the other hand, the test time is bigger.

Number of Tests: it is the number of tests you want to run. The final result will be an average measure.

On the middle, there is a sub-panel parameters configuration. The specific TOPP´s parameters could be set in this panel. Some of them are read-only because are determined by another parameters. There are eight in total:

Tau: it is the time in milliseconds of each stream. It is a read-only parameter. It is fixed by T_{in}. It is very important that the Tau values are not too big in order to obtain a more accurate measure because it has to be made in constant traffic conditions to get the ABw as effectively as possible.

T_{NI}: it is a security parameter that sets an amount of time between streams represented in milliseconds, in order to avoid the influence between one fleet on the other.

T_{in}: it is the time in microseconds between packets. It is a read-only parameter because it is fixed by R_{min} and Packet Size parameters.

Number of Packets: this value could be modified. The more packets you send, the better results you get, but the Tau value increases. The default value is set to 10.

Eta: this parameter set an amount of time between packets pairs. It represents a percentage of T_{in} in order to set the minimum T_{pp}.

Number of streams: the more streams you set, the better results you get because you increase the number of samples to get better results.

Packet Size: it is the amount of bytes it is sent into each packet. The optimal value of this parameter varies between 800 and 1500 bytes. But the application sets the maximum value in 1458 bytes in order to account the headers. The minimum packet size is fixed at 192 bytes.

Intrus: it is the intrusion level in Mbps. It is a read-only parameter. This parameter shows the amount of bits per second spent on each stream.

The last sub-panel is about the parameters accuracy. By this sub-panel, it could be set the measures accuracy.

RPOK: the packets received are OK. It is the percentage of packets that are included in the desired time. The optimal values for this parameters are between 50 and 60 %.

T_{in}ME: T_{in} Merge Error. It is the amount of time in percentage. The T_{in} received may vary. An optimal value may be between 8 and 12 %.

Results

The results are shown in the result panel, with the tab TOPP activated.

First, it shows the algorithm configuration, i.e. the values of the parameters selected for the measure.

Then, the results of each iteration are represented in Figure 79. These results demonstrate the shipping rate, the rate at which it is received, the fraction between the rate sent and received rate, number of packets arrived in the timeframe which is fixed by T_{in}ME and the percentage error of these packages.

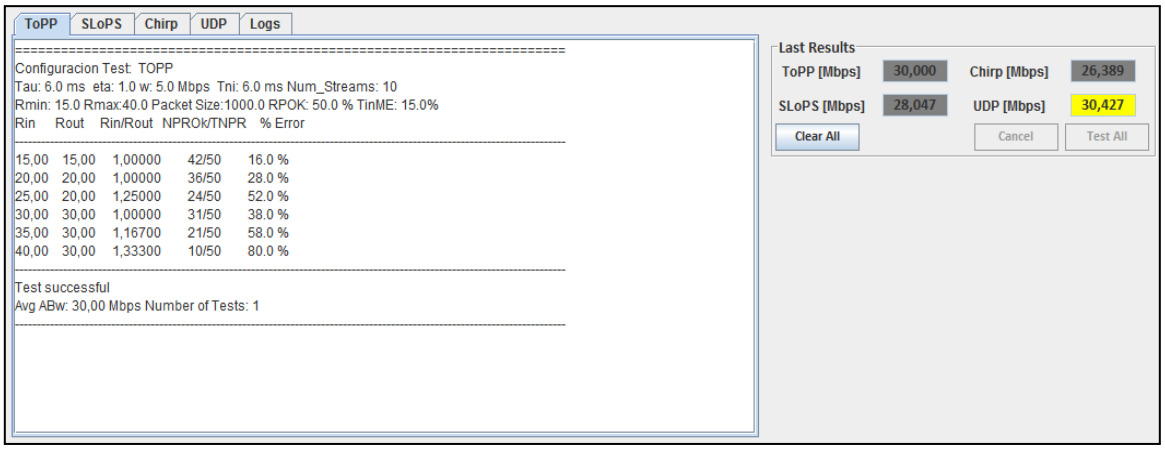

Figure 79. TOPP results

Also it could be seen the result in the last panel. The bigger of all is highlighted in yellow color.

1.4 SLoPS

Parameters

The Figure 80 shows the SLoPS panel with the corresponding structure, as is required to send packages.

As shown in the rigth picture, SLoPS is similar to TOPP, except that instead of changing $T_{in,}$ the packet size is changed on each iteration.

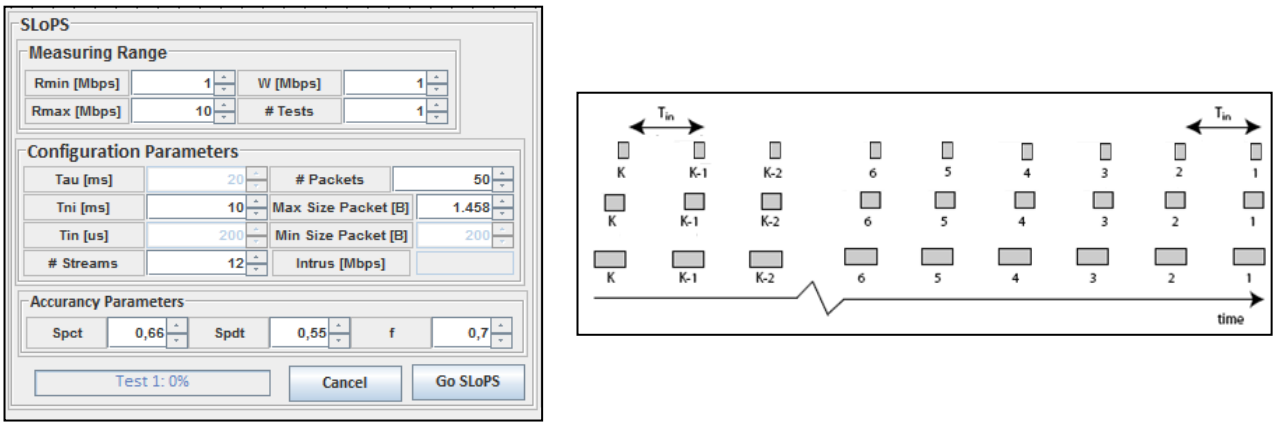

Figure 80. SLoPS configuration

On top of the SLoPS panel, as in TOPP, a measuring range sub-panel it is shown. There are the same parameters and the same meanings. This settings could be read in TOPP section.

On the middle, as in TOPP, there is a sub-panel parameters configuration. The specific SLOPS´s parameters could be set in this panel, some of them are read-only because are fixed by another parameters. There are eight in total:

Tau: it is the time in milliseconds of each stream. It is a read-only parameter. It is fixed by Tin. It is very important that the Tau values are not too big in order to obtain a more accurate measure because it has to be made in constant traffic conditions to get the ABw as effectively as possible.

T_{NI}: it is a security parameter that sets an amount of time between streams represented in milliseconds, in order to avoid the influence between one fleet on the other.

T_{in}: it is the time in microseconds between packets. It is a read-only parameter because it is fixed by R_{min} and Packet Size parameters.

Number of streams: the more streams you set, the better results you get because you increase the number of samples to get better results.

Number of Packets: this value could be modified. The more packets you send, the better results you get, but the Tau value increases. The default value is set to 50.

Maximum Packet Size: it is the maximum amount of bytes that can be sent per packet. The optimal parameters value is between 800 and 1500 bytes. But the application set the maximum value in 1458 bytes in order to account the headers. The less amount of bytes, the less range to test, because there is a minimum packet size limit: 192 bytes.

Minimum Packet Size: it is the minimum amount of bytes that can be sent on each packet. It is a read-only parameter because it is fixed by R_{max}.

Intrus: it is the intrusion level in Mbps. It is a read-only parameter. This parameter shows the amount of bits per second spent on each stream.

The last sub-panel is about the parameters accuracy. By this sub-panel, you could set the measures accuracy.

S_{pct}: Pairwise Comparasion Test, is one of a statistics parameter to check if a stream shows a increasing trend. PCT measures the fraction of consecutive OWD pairs that are increasing, and so $0 < S_{pct} < 1$. If the OWDs are independent, the expected S_{pct} value is 0.5. If there is a strong increasing trend, S_{pct} approaches one. According to some researches, the PCT metric reports *increasing trend* if $S_{pct} > 0{,}66$, *non-increasing trend* if $S_{pct} < 0{,}54$ and *ambiguous trend* otherwise.

S_{pdt}: Pairwise Different Test, is the other statistics parameter to check if a stream shows an increasing trend. PDT quantizes how strong is the start-to-end OWD variation, relative to the OWD absolute variations during the stream. Note that $-1 < S_{pdt} < 1$. If the OWDs are independent, the expected S_{pdt} value is zero. If there is a strong increasing trend, S_{pdt} approaches one. According to some researches, the PDT metric reports *increasing trend* if $S_{pdt} > 0{,}55$, *non-increasing trend* if $S_{pdt} < 0{,}45$ and *ambiguous trend* otherwise.

When one of the PCT and PDT metrics reports *increasing trend*, while the other is either *increasing* or *ambiguous*, the stream is characterized as type-I (for *increasing*). Similarly, when one metric reports *non-increasing* trend, while the other is either *non-increasing* or *ambiguous*, the stream is characterized as type-N (for *non-increasing*). If both metrics report *ambiguous*, or when one is *in-creasing* and the other is *non-increasing*, the stream is discarded.

f: if a fraction f of the N streams is type-N, the fleet does not show increasing trend and it could be inferred that the fleet rate is smaller than the ABw (R < A). It can happen, though, that less than N x f streams are type-I, and also less than N x f streams are type-N. f is set to 70% by default.

Results

As represented in TOPP, the first thing shown is the test configuration.

Then, the results of each iteration are displayed. In this case, the result is understood by the sending rate and the corresponding region, i.e., if it has been an increasing or decreasing trend in the path.

Finally, the final result is shown together with the number of tests that have been made.

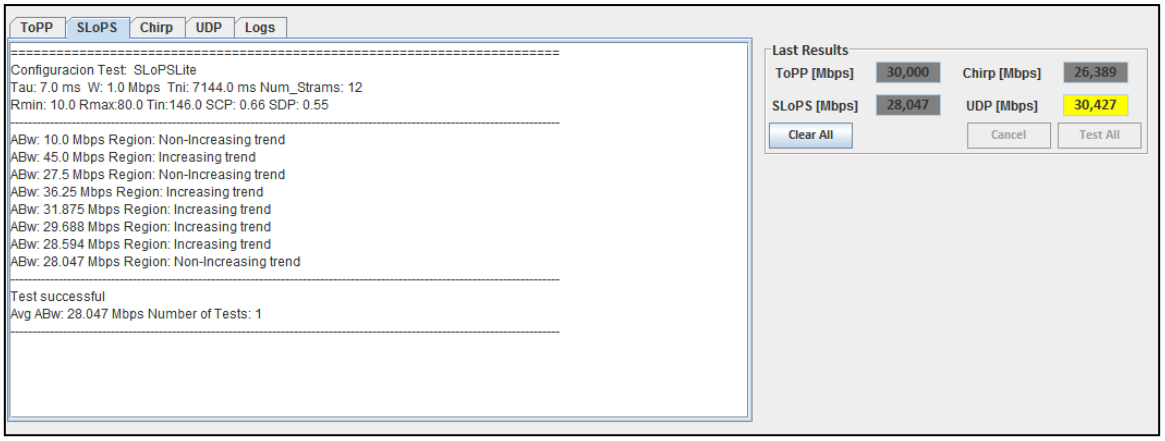

Figure 81. SLoPS results

Also it could be seen the result updates in the last panel, but it is not highlighted in yellow because it is not the maximum value.

1.5 PathChirp

Parameters

The Figure 82 shows the PathChirp panel with corresponding structure, as is required to send packages.

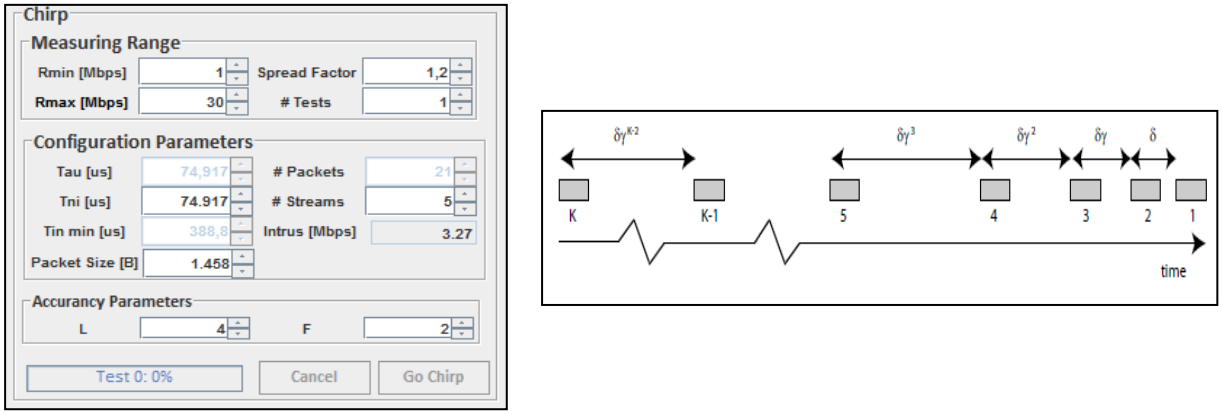

Figure 82. PathChirp configuration

On top of the PathChirp panel, in the same way that TOPP and SLoPS show. a measuring range sub-panel it is shown. There are three similar parameters with the same meanings, it could be read this in TOPP section, and there is one new,

Spread factor: sets the time gap between two consecutive packets as shown in Figure 82. In this figure, there is an indirect parameter, δ which is fixed by the fraction L/R_{max}, where L is the packet´s size.

On the middle, as in TOPP, there is a sub-panel parameters configuration. The specific PathChirp´s parameters could be set in this panel. Some of them are read-only because are determined by another parameters. There are seven in total:

Tau: it is the time in milliseconds of each stream. It is a read-only parameter. It is fixed by Tin. It is very important that the Tau values are not too big in order to obtain a more accurate measure because it has to be made in constant traffic conditions to get the ABw as effectively as possible.

T_{NI}: it is a security parameter that sets an amount of time between streams represented in milliseconds, in order to avoid the influence between one fleet on the other.

T_{in}: it is the time in microseconds between packets. It is a read-only parameter because it is fixed by R_{min} and Packet Size parameters.

Packet Size: it is the amount of bytes it is sent into each packet. The optimal value of this parameter varies between 800 and 1500 bytes. But the application sets the maximum value in 1458 bytes in order to account the headers. The minimum packet size is fixed at 192 bytes.

Number of Packets: this value is fixed by R_{max} value. It is a read-only parameter.

Number of streams: the more streams you set, the better results you get because you increase the number of samples to get better results.

Intrus: it is the intrusion level in Mbps. It is a read-only parameter. This parameter shows the amount of bits per second spent on each stream.

The last sub-panel is about the parameters accuracy. By this sub-panel, you could set the measures accuracy.

L: it is the excursion length threshold. This parameter determines a valid excursion length. The optimal values could be place between 4 and 6.

F: it is a decrease factor. F indicates at which point the OWD has decreased this factor. An optimal value could be between 1.5 and 2.5.

Figure 83 represents both concepts.

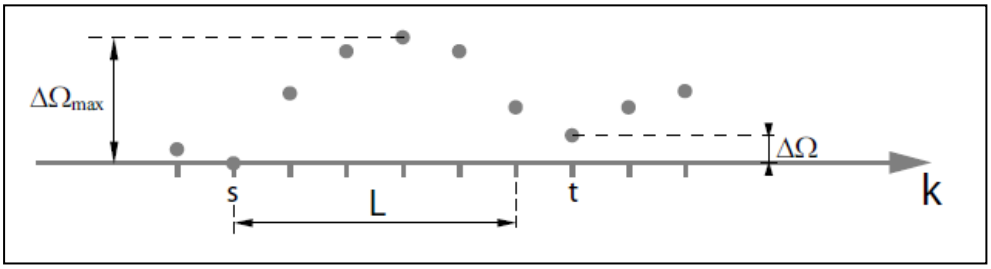

Figure 83. If ($\Delta\Omega_{max} > F\ \Delta\Omega$) and ($t - s \geq L$) then an excursion is detected with starting point s and ending point t.

Results

First, the initial configuration is showed as the same as in the previous two algorithms.

Next it is time to get the results. In this case, the algorithm performs one iteration only, which shows it is result and also the lost packets. Four tests are completed in the Figure 84.

Finally, the average result obtained in the tests performed is shown on it, and also the number of test.

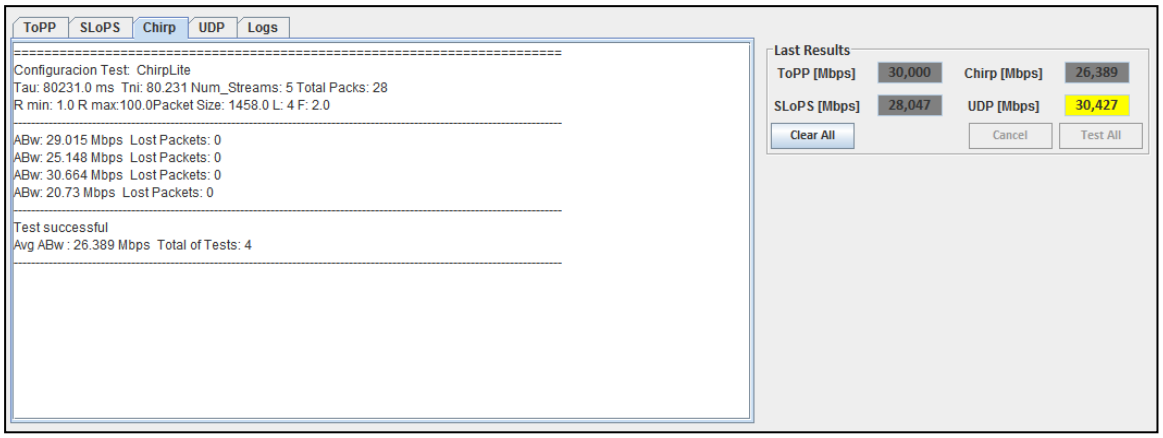

Figure 84. PathChirp results

Again, it updates the latest results panel.

1.6 UDPFile

Parameters

UDPFile is based on a fixed average rate of packets shipping between the server and the client. The ABw is estimated after measuring the ratio between the number of bytes received and the time spent on shipping. But there is no transmission time or delay. It is the fraction between the number of bytes received and the spent seconds.

Figure 85 shows the PathChirp panel with corresponding structure, as is required to send packages.

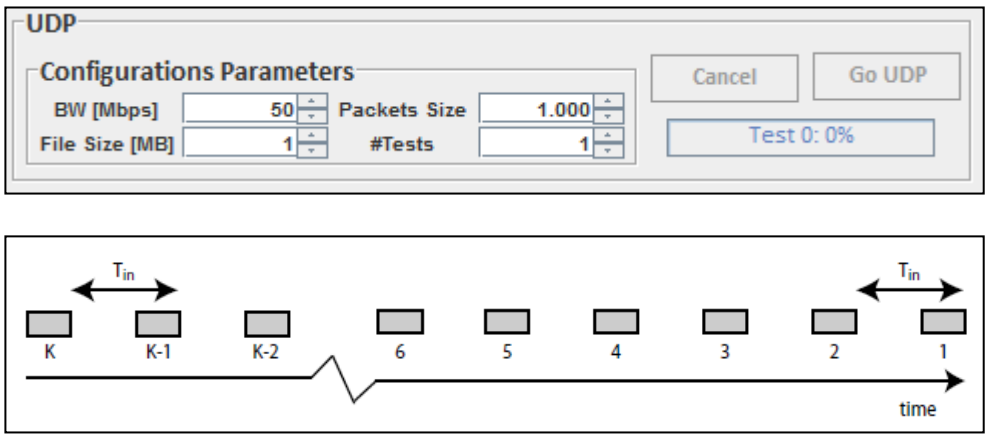

Figure 85. UDPFile configuration

There are four parameters,

BW: is the maximum rate to test. It is the average rate through which the sender ships packets. If it is necessary to measure X Mbps an upper value than X shall be set.

File Size: the total bytes that the sender ships.

Packet Size: it is the amount of bytes it is sent into each packet. The optimal value of this parameter varies between 800 and 1500 bytes. But the application sets the maximum value in 1458 bytes in order to account the headers.

Number of Tests: it is the number of tests that desired to run. The final result will be an average measure.

Results

As said before, the first part corresponds to the algorithm´s configuration.

Here, as in PathChirp that there is only one iteration this one comes along with the number of packets received and lost.

Finally, the total result with the number of tests performed are showed.

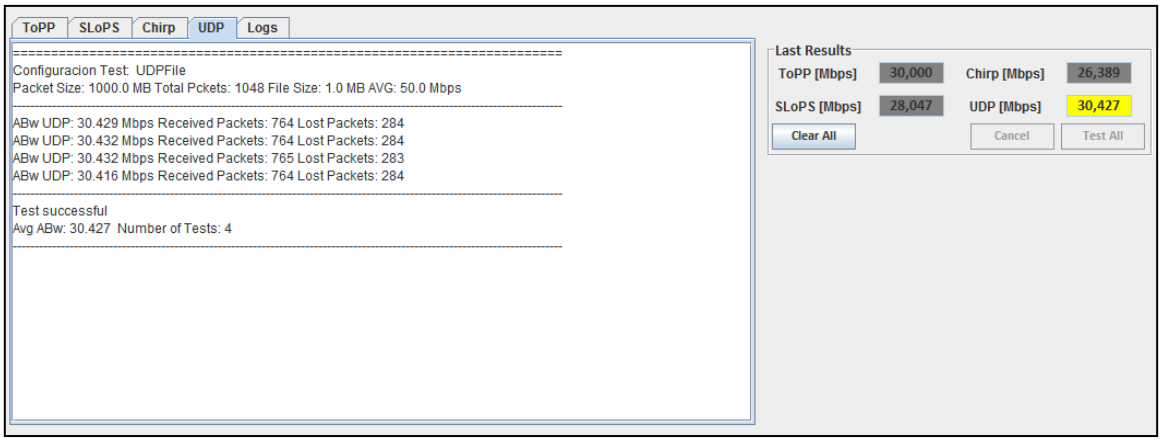

Figure 86. UDPFile results

It also updates the last panel results. In this case, is the highest value and is highlighted in yellow color.

1.7 Logs Tab

All test algorithms that have been implemented all summarized in this tab. It is similar to a log file.

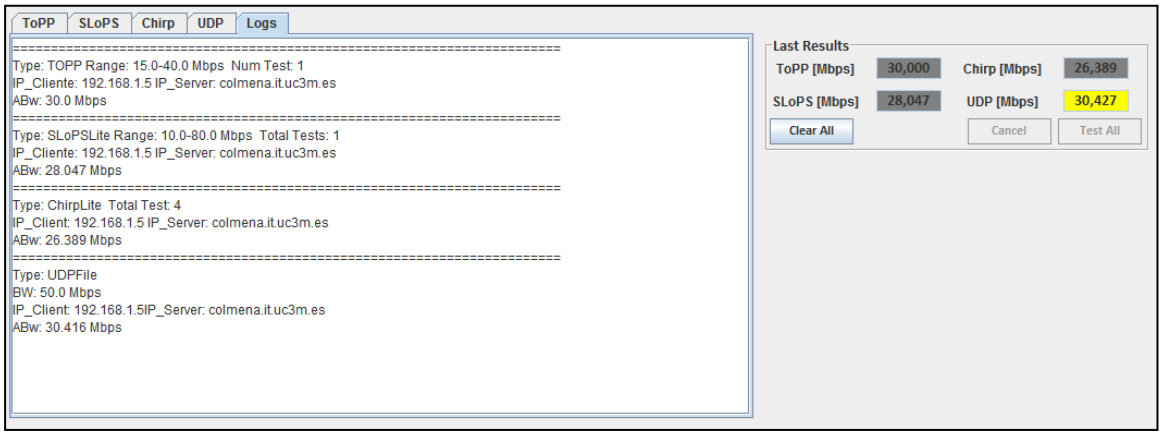

Figure 87. Log tab

Apéndice 2: WANem User´s Guide

This appendix will show a quick guide to install and use WANem.

2.1 Setting up WANem

WANem is distributed in the form of a bootable CD with Linux Knoppix O/S. This CD comes with WANem preinstalled.
No there are no installation steps. When an i386 architecture based PC is booted with the PC WANem is
ready for use.

2.1.1 How to start WANem

The distribution is in the form of a bootable CD. No installation is required. Just insert the CD into any i386 PC/Server and reboot using the CD. The PC will boot up in the Knoppix Linux OS. We will call this the WANem PC. After the PC boots up, it automatically starts the IP address configuration screen.

Following are the general steps to be followed after WANem boots up

1. You get a Knoppix screen with a prompt in the bottom left called “boot:” Just press “Enter”
2. After Knoppix boots up a question is prompted if you want to configure all the network interfaces of the PC using dhcp. If the PC is connected to a dhcp network, then it is recommended to enter “y” and move on to step 5.
3. Otherwise you will be automatically taken to a network setup screen.
4. Setup the IP address for the Ethernet interface (most likely eth0). Call this “wanemip”. You have to select this interface and set up the IP address, network mask and default gateway. See Figure 88.

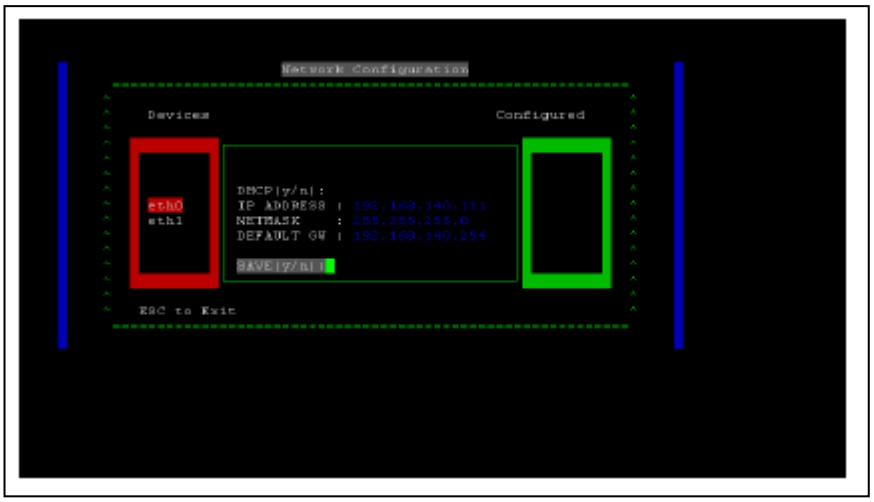

Figure 88. Setup IP address for Ethernet interface [WEM]

5. WANem will prompt you to enter the password for user “perc”. It will also prompt you to re-enter the password. Using this user id and password will allow you to remote login to the WANem PC with programs like putty or ssh.
6. Setup the routes between the two end hosts (say client and server) such that their packets are routed via the WANem PC when the two communicate with each other. For more details refer to section “How to make packets go via WAN emulator”.
7. From another Windows PC on the network open Internet Explorer / Firefox and type the URL http://wanemip/WANem. [Ensure that your screen resolution is set to 1024 x 765 for best viewing.]
8. Now, WANem is ready for use.

2.2 Using WANem Graphical user interface

In the WANem start page you will see five options: About, WANalyzer, Basic Mode, Advance Mode, Help.

2.2.1 WANalyzer

If the WANalyzer option is selected then you will see the GUI as shown in Figure 89. The popup window is the result window and initially not shown. To measure the WAN characteristics between the WANem box and the remote machine, one needs to enter the IP address of the remote machine. If the remote machine is reachable then WANalyzer will measure the WAN characteristics and produce the results as shown in Figure 90.

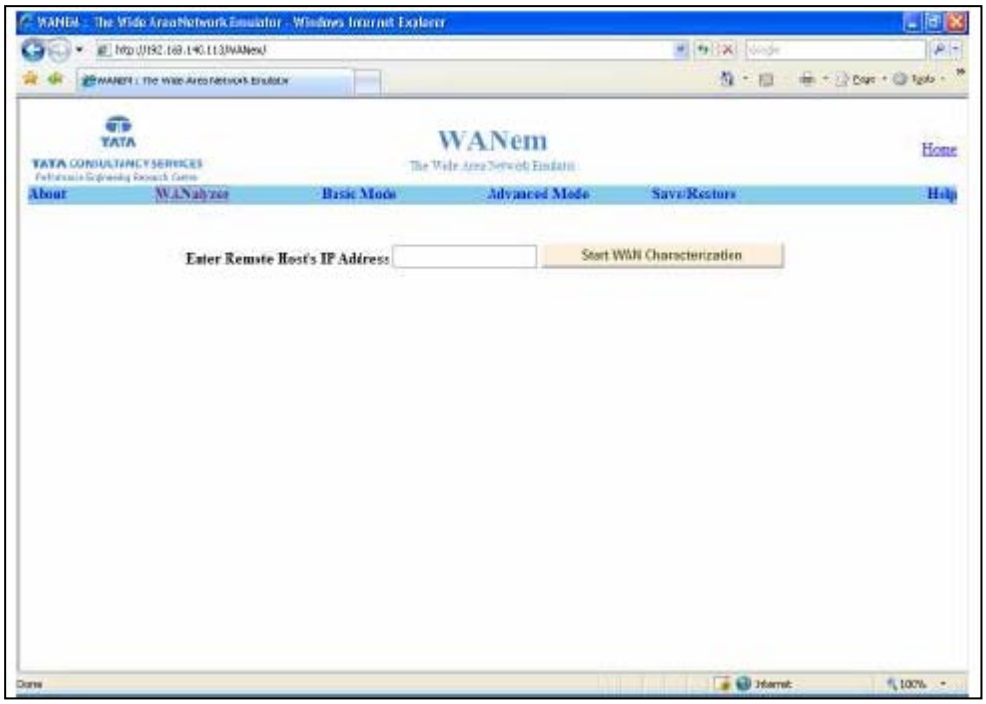

Figure 89. WANalyzer GUI [WEM]

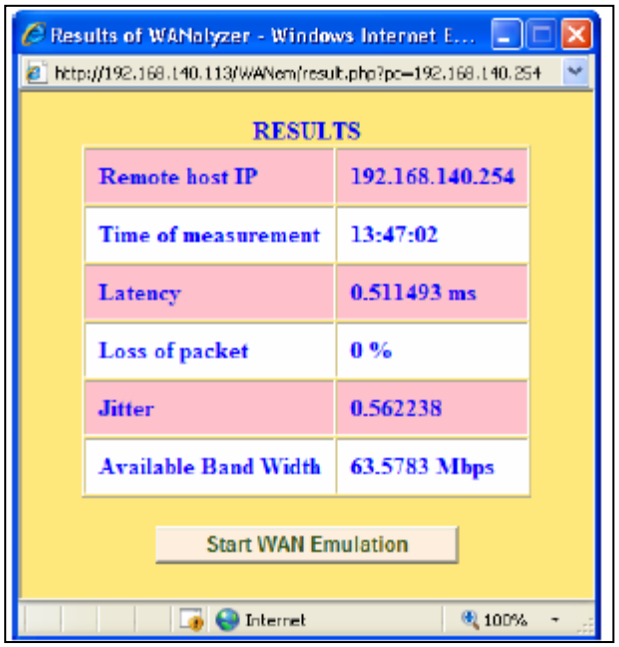

Figure 90. WANalyzer GUI with result window [WEM]

2.2.2 Basic Mode

If the basic mode is selected then only one set of network characteristics can be applied for each network interface. Each such set is called a rule set. In the basic mode each rule set will allow the user to specify only the following network characteristics

- Bandwidth
- Latency

Illustrated in Figure 91 is a screenshot of the WANem GUI in basic mode with one network interface.

Note that - user can either choose bandwidth or specify bandwidth.

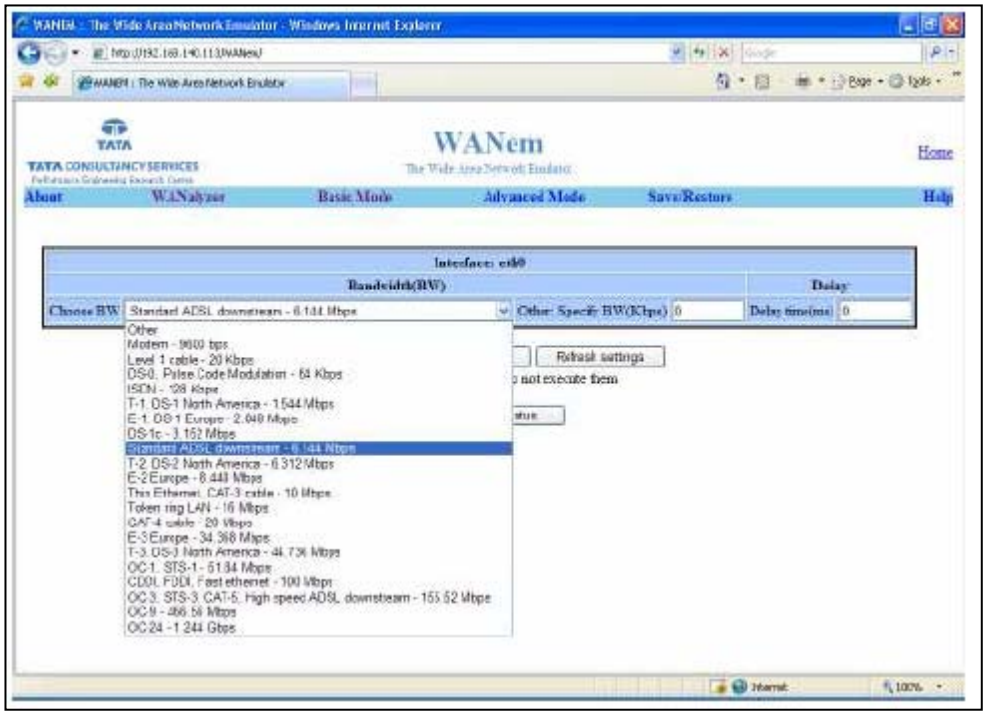

Figure 91. WANem GUI - Basic Mode [WEM]

2.2.3 Advanced Mode Operation

If you want to use the advanced mode select the correct network interface in advanced mode and click the “Start” button. Typically eth0 is the default Ethernet network interface you want to use. You can confirm this with your system/network administrator. You get a screen as shown in Figure 92. WANem GUI - Advance Mode [WEM]

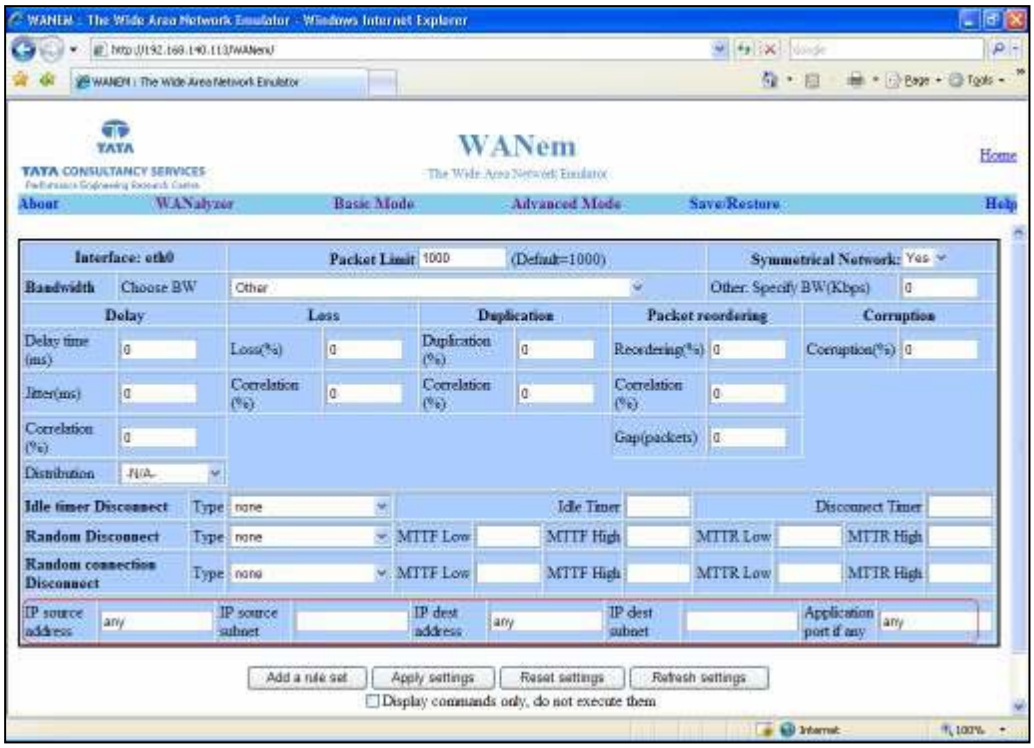

Figure 92. WANem GUI - Advance Mode [WEM]

1. The above screen shows the form for one rule set. All fields barring the ones in the last row represent various network characteristics. Fill in the relevant network characteristics.
2. The last row (encircled) in the form starting with “IP source addr” determines the packets to which the rules apply. There are three possibilities here

   a) You want the same rule set to apply to any packet that passes via WANem. In this case you can leave this row as it is.
   b) You want the rule set to apply to all traffic between two end hosts (regardless of whether they are client or server). Then fill in the specific IP addresses or the 2 hosts. You can set the subnets to 32. Leave the “Application port” to “any”.
   c) You want the rule set to apply to all traffic between two end hosts which are client and server for a given application. Then fill in the specific IP addresses of the client in “IP source addr”. And the address of the server in “IP destination addr”. You can set the subnets to 32. Set the “Application port” to the server port

   .
3. To add one more rule set then click “Add a rule set” and repeat from step 2.
4. Click “Apply Settings” to start WANem.
5. Click “Refresh Settings” immediately to ensure that the settings have taken effect.

Saving all your network characteristics for later reuse is a useful feature to have. This will be supported in the next release of WANem.

2.3 How to make packets go via WAN emulator

2.3.1 Using Multiple NIC cards on WANem PC

In this situation the WANem PC has two active network interface cards. In other words both Network cards are connected to a switch on the LAN. This can be used to advantage as it will increase the capacity of WANem to handle more packets per second. The idea is distribute end hosts to accross interface cards in the WANem PC.

Note that – this will be useful also if client and server are in different subnets.

This is illustrated in the following figure. Since each end host is assigned to one interface card Symmetrical private WANs will be composed of two rule sets – one for either interface.

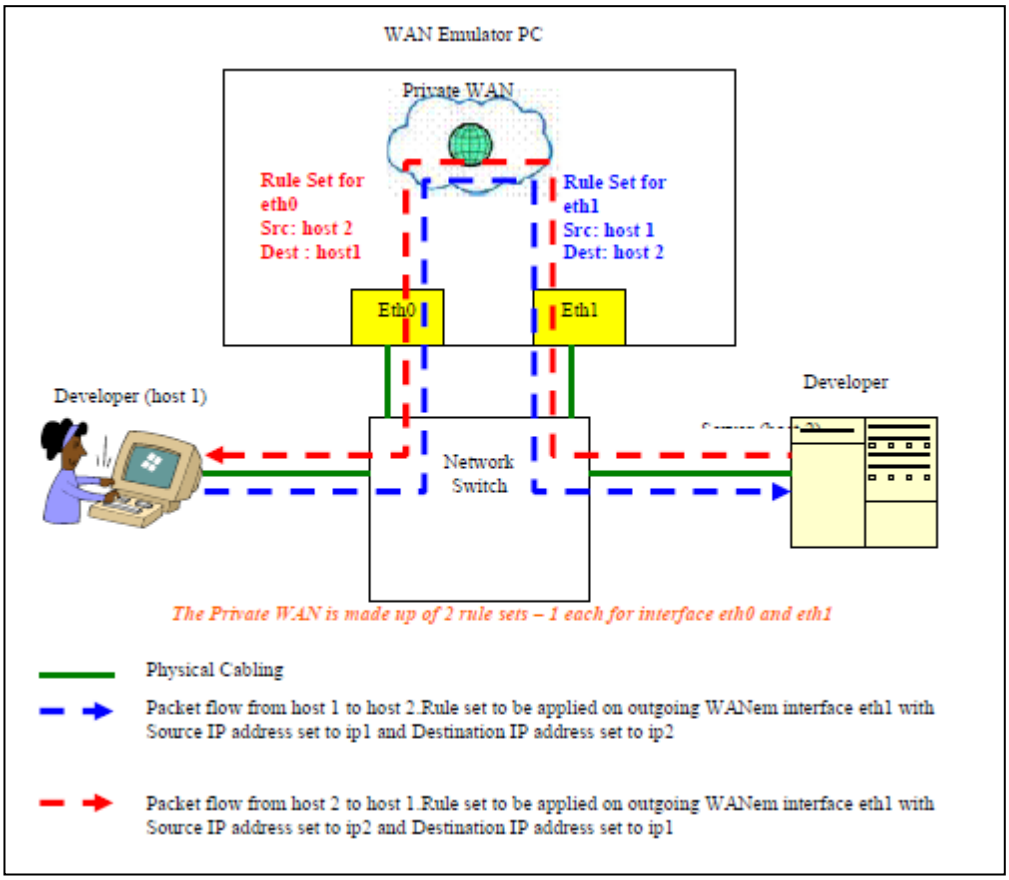

Figure 93. Routing packets through WANem PC with two NICs [WEM]

It is important to ensure that there is one IP address assigned to each network card. As before assume that the IP address for host 1 is ip1 and that for host 2 is ip2. This is normally done during when the WANem PC boots up. If this has not been done the reset command can be used to set IP addresses. In order to configure WANem in this scenario.

1. If network addresses have not been set on all the network interfaces, then run command “reset” on the WANem PC console. Otherwise use the status command to know the IP address of each interface. The output of this command will list all the interfaces. For e.g. If there are two active Ethernet interfaces then there will eth0 and eth1.
2. Select an ip address for eth0. Call this wanemip1. Also enter default gateway and network mask4. See Figure 88. Setup IP address for Ethernet interface [WEM]
3. Select an ip address for eth1. Call this wanemip2. Also enter default gateway and network mask. See Figure 88. Setup IP address for Ethernet interface [WEM]
4. Enter yes to “Save and quit” prompt
5. Assign host 1 to eth0 on WANem PC by running the command “assign <ip1> eth0”
6. Assign host 2 to eth1 on WANem PC by running the command “assign <ip2> eth1”
7. On host 1 add a route to host2 using IP address wanemip1. This will cause packets originating from host 1 and destined to host 2 to enter WANem PC via interface eth0. The commands required for doing are illustrated for different OSes in the next section called Changing routes.
8. Similarly on host 2 add a route to host1 using IP address wanemip2. This will cause packets originating from host 2 and destined to host 1 to enter WANem PC via interface eth1.
9. Using the WANem GUI select interface eth1. Add a new rule set with source IP address set to ip1 and Destination IP address set to ip2. All the network characteristics set will apply for packets flowing from host 1 to host 2. This is because the WANem network characteristics are applied only when the packet is about to exit the WANem PC via the outgoing interface which in this case is eth1.
10. Using the WANem GUI select interface eth0. Add a new rule set with source IP address set to ip2 and Destination IP address set to ip1. All the network characteristics will apply for packets flowing from host 1 to host 2. Reasoning is similar to the point above.

Suppose there are 2 more hosts, host 3 (with IP address ip3) and host 4 (with IP address ip4) which require WAN emulation then it will be a good idea to assign host 3 to WANem PC eth0 and host 4 to WANem PC eth1. With this strategy repeat the steps above.

2.3.2 Changing Routes

In the following examples we will setup the network traffic between two hosts to flow via WANem. Following rules will be used for illustrating routing change examples.

- IP address of host 1 is 192.168.140.12
- IP address of host 2 is 192.168.140.14
- IP address of WANem PC is 192.168.140.20

It is assumed that the user is logged in to the machines using Administrator or root privileges.

Example 1: Both host1 and host 2 run the Windows O/S

1. Run this command host 1 “route add 192.168.140.14 mask 255.255.255.255 192.168.140.20.

2. Run this command host 2 “route add 192.168.140.12 mask 255.255.255.255 192.168.140.20.

On Table 27 show a listing of commands for changing routes where the end hosts run different combinations operating systems.

Example Scenario	Host 1 operating system	Change route to WANem command on Host 1	Host 2 operating system	Change route to WANem Command on Host 2
Two Windows Hosts	Windows	route add 192.168.140.14 mask 255.255.255.255 192.168.140.20	Windows	route add 192.168.140.12 mask 255.255.255.255 192.168.140.20
Windows and Linux	Windows	route add 192.168.140.14 mask 255.255.255.255 192.168.140.20	Linux	route add –host 192.168.140.12 netmask 0.0.0.0 gw 192.168.140.20
Linux and IBM AIX	Linux	route add –host 192.168.140.14 netmask 0.0.0.0 gw 192.168.140.20	IBM AIX	route add 192.168.140.12 192.168.140.20
IBM AIX and Solaris	IBM ATX	route add 192.168.140.12 192.168.140.20	Solaris	route add 192.168.140.12 192.168.140.20

Table 27. List of commands for changing routes [WEM]

Apéndice 3: pseudo-código SLoPS

En la Figura 94 se muestra el código con el que SLoPS decide la tasa a enviar en función de si se ha producido una tendencia creciente o decreciente. Como se ha mencionado en el capítulo 4, este código se ha modificado para evitar problemas a la hora de encontrar una región gris. Simplemente, la parte correspondiente a la región gris no se utiliza nunca.

```
Rate adjustment algorithm:
{
/* Initially: G^min = G^max = 0 */

    if (R(n) < A) /* Non-increasing trend */
        R^min = R(n);
        if (G^min > 0)
            R(n+1) = (G^min + R^min)/2;
        else
            R(n+1) = (R^max + R^min)/2;
    else
    if (R(n) > A) /* Increasing trend */
        R^max = R(n);
        if (G^max > 0)
            R(n+1) = (R^max + G^max)/2;
        else
            R(n+1) = (R^max + R^min)/2;
    else
        /* Grey-region */
        if (G^min == G^max == 0)
            G^min = G^max = R(n);
        if (G^max ≤ R(n))
            G^max = R(n);
            R(n+1) = (R^max + G^max)/2;
        else if (G^min > R(n))
            G^min = R(n);
            R(n+1) = (R^min + G^min)/2;

    /* Termination conditions */
    if (R^max − R^min ≤ ω || (G^min − R^min ≤ χ && R^max − G^max ≤ χ))
        return (R^min, R^max);
}
```

Figura 94. Pseudo-código de SLoPS [JC02]

Apéndice 4: pseudo-código PathChirp

En la Figura 95 se muestra el pseudo-código del algoritmo *pathChirp*. Como se puede apreciar, se ven dos métodos, *excursión* y *estimate_D*. El primero calcula la longitud de una *excursión* producida y el segundo decide si es válida o no en función de esa longitud.

Este algoritmo no ha sufrido ninguna modificación, con lo que el código implementado es prácticamente idéntico.

```
pathChirp Algorithm
 procedure estimate_D(q){
   /* q denotes the vector of a single chirp train's
    queuing delays */
   for (k = 1 to N − 1) E_k = 0;  /*initialize*/
   i = 1; /* Denotes current packet number */
   l = N − 1; /* N=number of chirp packets*/
   while(i ≤ N − 1){
     if (q_i < q_{i+1}){
       j = excursion(q,i,F,L)
       choose case(j):
         Case(a): (j > i) and (j ≤ N)
           for (s = i to j − 1)
             if (q_s < q_{s+1}) E_s = R_s;
         Case(b): j = N + 1
           for (s = i to N − 1) E_s = R_i;
           l = i;
       /* end choose case */
       if (j = i) j = j + 1;
       i = j;
       } /* end if */
     else
       i = i + 1;
   } /* end while*/
   D = 0;
   for (i = 1 to N − 1){  /*computing D*/
     if (E_i == 0)
       D+ = R_l Δ_i; /* Case (c) */
     else
       D+ = E_i Δ_i;
   }; /* end of for loop */
   D = D/∑_{1≤i≤N−1}(Δ_i);
   return D;
 }

procedure excursion(q,i,F,L){
 j = i + 1;
 max_q= 0;
 while((j ≤ N) and (q(j) − q(i) > max_q/F))
   {
    max_q=maximum(max_q,q(j) − q(i));
    j = j + 1;
   }
 if ((j ≥ N)) return j;
 if (j − i ≥ L)
    return j;
 else
    return i;
}
```

Figura 95. Pseudo-código de PathChirp [Rib+03]

Glosario

ABw	Available Bandwidth
ABwE	Available Bandwidth Estimated
ADSL	Asynchronous Digital Subscriber Line
API	Application Program Interface
ATM	Asynchronous Transfer Mode
BTC	Bulk Transport Capacity
CBR	Constant Bit Rate
CPU	Central Processing Unit
DP	Direct Probing techniques
FIFO	First In Firs Out
GUI	Graphical User Interface
IANA	Internet Assigned Numbers Authority
IGI	Initial Gap Increases
IP	Internet Protocol
ItP	Iterative Probing techniques
JVM	Java Virtual Machine
LAN	Local Area Network
MTU	Maximum Transmission Unit
NIC	Network Interface Card
OWD	One-Way Delay

PGM	Probe Gap Model
PRM	Probe Rate Model
RAM	Random Access Memory
RFC	Request for Comments
RTT	Round Trip Time
SLoPS	Self-Loading Periodic Streams
$\mathbf{S_{pct}}$	Parwise Compare Trend
$\mathbf{S_{pdt}}$	Parwise Difference Trend
TCP	Transport Control Protocol
TOPP	Train of Packets Pairs
$\mathbf{T_x}$	Tráfico Cruzado
UDP	User Datagram Protocol
URL	Uniform Resource Locator